Ada-Diffuser: Latent-Aware Adaptive Diffusion for Decision-Making

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent work has framed decision-making as a sequence modeling problem using generative models such as diffusion models. Although promising, these approaches often overlook latent factors that exhibit evolving dynamics, elements that are fundamental to environment transitions, reward structures, and high-level agent behavior. Explicitly modeling these hidden processes is essential for both precise dynamics modeling and effective decision-making. In this paper, we propose a unified framework that explicitly incorporates latent dynamics inference into generative decision-making from minimal yet sufficient observations. We theoretically show that under mild conditions, the latent process can be identified from small temporal blocks of observations. Building on this insight, we introduce `Ada-Diffuser`, a causal diffusion model that learns the temporal structure of observed interactions and the underlying latent dynamics simultaneously and leverages them for planning and control. With a proper modular design, `Ada-Diffuser` supports both planning and policy learning tasks, enabling adaptation to latent variations in dynamics, rewards, and even recovering hidden action variables from action-free demonstrations. Extensive experiments on locomotion and robotic manipulation benchmarks demonstrate the model's effectiveness in accurate latent inference, long-horizon planning, and adaptive policy learning.

1 Introduction

Learning and planning in partially observable environments is a fundamental challenge in building intelligent agents [1]. Recent work on casting decision making as a generative modeling problem, taking advantage of powerful models such as transformers [2–4], diffusion models [5–13], and vision language action models [14–22], achieving impressive results in a wide range of tasks. However, these methods often fail to account for hidden latent variables and their temporal dynamics, factors that are prevalent in real-world settings such as robotics [23], autonomous driving [24], healthcare [25, 26], and economics [27]. Ignoring such latent processes can result in suboptimal decision-making, particularly when the observational data does not provide full coverage of the latent factors underlying the environment's dynamics [28–36, 30].

Early works address partial observability in reinforcement learning (RL) and imitation learning (IL) by encoding historical observations and actions into belief states or latent embeddings, which represent a distribution over the underlying latent state [1, 37–41, 36, 42, 43, 28, 44–46]. Policy optimization or planning is then carried out based on these inferred belief states. However, learning such representations often requires access to the historical trajectories or data from a diverse set of environments. This can be prohibitively expensive, particularly in high-dimensional state or action spaces, posing challenges for integrating these methods into modern generative decision-making models, which typically prioritize scalability. Can we *identify* the latent factors that govern environment dynamics and rewards, and integrate them into *scalable* generative decision-making

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

models to enable adaptive planning and policy learning, using only *minimal observations*, while *preserving theoretical guarantees*?

In this paper, we pursue this goal by addressing two fundamental questions. First, what is the *minimum* set of observations required, *in principle*, to reliably identify the latent factors that govern the environment? Second, how can *latent identification* be effectively incorporated into generative models (e.g., diffusion models) to enable adaptive planning and policy learning? To answer the first question, we theoretically show that, under mild conditions, the latent factors at the time step t can be block-wise identified using only four surrounding observable measurements (i.e., state-action trajectories) within a small temporal window. This identification result implies that a small temporal block is sufficient to infer the latent factors in observational RL trajectories in an *online* manner.

To answer the second question, we propose `Ada-Diffuser`, an *autoregressive* diffusion framework with latent identification from temporal blocks, designed to model the data generation process of RL trajectories influenced by latent factors. To reflect the autoregressive nature of sequential decision making, we introduce a *causal denoising schedule* that aligns the denoising steps with the underlying causal structure, drawing inspiration from recent advances in autoregressive diffusion models [47–50]. For temporal-block-wise latent identification, during training, we employ a *denoise-then-refine* procedure that iteratively alternates between denoising the observations and refining latent estimates. This enables `Ada-Diffuser` to jointly learn a structured representation of latent variables and the corresponding observational distribution. At inference time, `Ada-Diffuser` generates actions and states while estimating latent variables in an online fashion. Since states and actions are conditioned on the latent factors, we employ a *zig-zag sampling* scheme that alternates between sampling state-action pairs and updating latent variables, ensuring consistency between generated sequences and their underlying latent dynamics.

`Ada-Diffuser` provides a unified generative framework for sequential decision-making from demonstrations. It is applicable to both planning and policy learning tasks by conditioning on different types of observations and adapting the conditional generative process accordingly. The framework is flexible and can accommodate various forms of latent, including ones that influence dynamics, rewards, or even represent high-level latent actions. Importantly, even in environments without explicitly designed latent variables, the block-wise latent identification mechanism improves generative modeling by implicitly capturing structured temporal dependencies. This mechanism functions as a form of Bayesian filtering, enabling principled handling of uncertainty and temporal abstraction within the generative process. We demonstrate that `Ada-Diffuser` can be flexibly applied across a wide range of tasks, including simulated locomotion, robotic control and manipulation, accurately identifying latent factors and improving downstream policy learning and planning.

In summary, our main contributions are threefold: (1) We establish sufficient conditions under which latent factors influencing environment dynamics and rewards can be identified from short temporal windows of RL trajectories, without requiring full trajectory access or multi-environment data. (2) We develop `Ada-Diffuser`, a generative model that integrates temporal block-wise latent inference with causal diffusion modeling to capture structured dependencies and enables joint modeling of latent variables and observable sequences in decision-making. (3) `Ada-Diffuser` can be adapted to a wide range of decision-making tasks by conditioning on different types of observation. We empirically show the improved performance on various planning and control tasks.

2 Background and Related Work

In this section, we provide background and related work on diffusion-based decision-making. Additional discussions are provided in Appendix E, including related work on (1) learning latent belief states in POMDPs [1, 37–40, 51, 36, 30], particularly in the context of transfer, meta, and nonstationary RL/IL [28, 45, 43, 52, 53, 29, 45], and (2) autoregressive diffusion models [54–56, 47–50, 57].

Recent advances use diffusion models as planners and policy for both RL and IL. *I. Diffusion Planner*: Diffusion-based planning leverages generative models to sample future state-action trajectories from a given state, using guidance techniques [58, 59] to encourage desirable properties such as high expected rewards. Taking Denoising Diffusion Probabilistic Models (DDPM [60])-based approaches as an example, these methods learn a generative model over expert trajectories $\tau = \{(\mathbf{s}_0, \mathbf{a}_0), \dots, (\mathbf{s}_T, \mathbf{a}_T)\}$ by modeling a forward-noising process: $q(\mathbf{x}^t \mid \mathbf{x}^{t-1}) = \mathcal{N}(\mathbf{x}^t; \sqrt{\alpha_t}\,\mathbf{x}^{t-1}, (1-\alpha_t)\mathbf{I})$, and a parameterized denoising model $p_\theta(\mathbf{x}^{t-1} \mid \mathbf{x}^t)$ to reverse the process. Here, the superscript t denotes diffusion steps, T denotes the planning horizon, $\mathbf{x}^0$ is a clean subsequence sampled from the expert

trajectory τ, and α_t controls the variance schedule at diffusion step t. During inference, trajectories are generated by starting from Gaussian noise and iteratively denoising through the learned reverse process. This generation can be optionally conditioned on the initial state or other guidance signals $\mathbf{y}$ (e.g., goals, rewards): $\hat{\tau} \sim p_\theta(\tau \mid \mathbf{s}_0, \mathbf{y})$. *II. Diffusion Policy*: In contrast to diffusion planners, Diffusion Policy methods directly parameterize the policy $\pi_\theta(a \mid s)$ using diffusion models. For example, Diffusion Policy [7] uses a diffusion model to generate multi-step actions with expressive multimodal distributions. DPPO [8] extends this idea by modeling a two-layer MDP structure, which enables fine-tuning of diffusion-based policies in RL settings. Another line of work uses diffusion models to parameterize the policy networks for only the single current step [9–12]. `Ada-Diffuser` can generally accommodate both diffusion planner and policies within the same framework.

3 Latent Identification in POMDP

In this section, we seek to formally model the structure of the decision-making system by answering the following questions. First, where do the latent factors reside, and how do they influence the observable variables such as states, actions, and rewards? Second, can they be identified from demonstration data alone? We model the system that extends the standard MDP to include unobservable, time-varying latent variables that affect both the transition dynamics and the reward function. This model generalizes the contextual MDP by allowing the context to evolve stochastically over time. We then formalize the data generation process under this model using structural causal models (SCMs) [61]. Finally, we present theoretical results that characterize the minimal observational requirements for identifying the latent variables.

3.1 Latent Contextual POMDP with Time-Dependent Context

We model the latent factors using a general contextual MDP framework, where the context itself evolves over time. Formally, we define a latent time-varying contextual MDP as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{C}$ is the latent context space, $\mathcal{T}(\mathbf{s}_t \mid \mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{c}_t)$ is the transition distribution, $\mathcal{R}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c}_t)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The latent context $\mathbf{c}_t \in \mathcal{C}$ follows a time-dependent (possibly stochastic) process: $\mathbf{c}_t \sim p(\mathbf{c}_t \mid \mathbf{c}_{t-1})$, and is unobserved during training and inference. The agent only observes trajectories $\tau = \{(\mathbf{s}_0, \mathbf{a}_0), \ldots, (\mathbf{s}_T, \mathbf{a}_T)\}$, and infers the latent context $\mathbf{c}_t$ from the observational data. This is naturally relevant to several MDP models, including (dynamic) hidden parameter MDPs [62, 63, 29], Bayes-adaptive MDPs [64, 65, 28], and factored MDPs [66]. A full comparison and analysis is given in Appendix C.

Given trajectories generated under this model, we can describe the data generation process using the SCMs. Without the loss of generality, we consider the setting where an expert policy π is assumed to generate the actions, as is standard in learning from demonstration data. The data generation process can therefore be expressed as (l.h.s. Fig. 1):

$$
\begin{aligned}
&\textbf{Latent Dynamics:} && \mathbf{c}_t = h(\mathbf{c}_{t-1}, \eta_t), \\
&\textbf{State Transitions:} && \mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{c}_t, \epsilon_t), \\
&\textbf{Action Generation:} && \mathbf{a}_t = \pi(\mathbf{s}_t, \mathbf{c}_t), \\
&\textbf{Reward Function:} && r_t = g(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c}_t, \delta_t),
\end{aligned}
$$

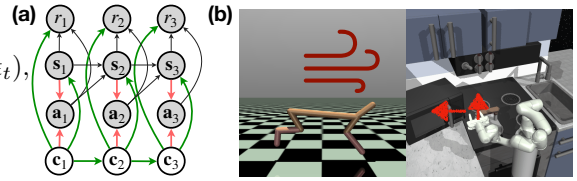

Figure 1: (a) SCM of the Latent Contextual POMDP. Gray/white nodes are observed/latent variables; green/red edges represent transitions driven by latents/expert policies, respectively. (b) Examples where latents influence either dynamics or rewards (affecting optimal actions).

where η_t, ϵ_t, and δ_t denote i.i.d. exogenous noise variables. Fig. 1(a) shows the graphical model. Fig. 1(b) illustrates examples where latent factors on dynamics (e.g., external wind in locomotion) and rewards (e.g., varying target objects in robot control) influence optimal decisions.

3.2 Identifiability of Latent Factors with Minimal Measurements

To learn accurate dynamics and make reliable decisions, it is essential that the underlying latent factors influencing the environment are identifiable with observational data. We present theoretical results that characterize the minimal number of consecutive observations required for the identifiability of the latent variables, under a set of mild and natural assumptions.

Assumption 1 (First-order Markov and limited feedback)**.** *We consider the following conditions:*

i. First-order Markov:

$$P\left(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{c}_t \mid \mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{c}_{t-1}, \boldsymbol{\omega}_{<t-1}\right) = P\left(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{c}_t \mid \mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{c}_{t-1}\right),$$

where $\boldsymbol{\omega}_{<t-1} = \{\mathbf{s}_{t-2}, \ldots, \mathbf{s}_1, \mathbf{a}_{t-2}, \ldots, \mathbf{a}_1, \mathbf{c}_{t-2}, \ldots, \mathbf{c}_1\}$.

ii. Limited feedback: $P\left(\mathbf{s}_t, \mathbf{a}_t, r_t \mid \mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{c}_t, \mathbf{c}_{t-1}\right) = P\left(\mathbf{s}_t, \mathbf{a}_t, r_t \mid \mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{c}_t\right)$.

Assumption 1(i) is naturally satisfied under our setting described in Section 3.1. Assumption 1(ii) specifies that the latent variable has no delayed effect on the state or action, implying that its influence is immediate. For brevity, we define the temporal state-action pairs $\mathbf{x}_t = [\mathbf{s}_t, \mathbf{a}_t]$ with the support $\mathcal{X}_t$.

Assumption 2 (Distributional Variability)**.** *There exist observed state and action variables $\mathbf{x}_t$ such that for any $\mathbf{x}_t \in \mathcal{X}_t$, there exists a corresponding $\mathbf{x}_{t-1} \in \mathcal{X}_{t-1}$ and a neighborhood $\mathcal{N}^r$ around $(\mathbf{x}_t, \mathbf{x}_{t-1})$ satisfying that, for all $\mathbf{x}_{t-2} \in \mathcal{X}_{t-2}$, $\mathbf{x}_{t-1} \in \mathcal{X}_{t-1}$, $\mathbf{x}_t \in \mathcal{X}_t$, and $\mathbf{x}_{t+1} \in \mathcal{X}_{t+1}$, the following conditional distribution operators are injective: (i) $L_{\mathbf{x}_{t-2}|\mathbf{x}_{t+1}}$, (ii) $L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}$, and (iii) $L_{\mathbf{x}_t|\mathbf{x}_{t-2},\mathbf{x}_{t-1}}$, where the conditional operator L represents transformations at the distribution level, that is, how one probability distribution is pushed forward to another [67].*

Assumption justification. Conceptually, the injectivity of these operator L implies that different inputs induce different output distributions, thus imposing a minimal condition on distributional variability. In RL systems, this condition is naturally satisfied in most stochastic environments where transitions produce sufficient diversity across different states and actions. The assumption also aligns with the standard conditions in identification theory, particularly in works that use spectral methods and latent variable models [68, 69].

Assumption 3 (Uniqueness of Spectral Decomposition)**.** *For any $\mathbf{x}_t \in \mathcal{X}_t$ and any $\bar{\mathbf{c}}_t \neq \tilde{\mathbf{c}}_t \in \mathcal{C}_t$, there exists a $\mathbf{x}_{t-1} \in \mathcal{X}_{t-1}$ and corresponding neighborhood $\mathcal{N}^r$ satisfying Assumption 2 such that, for some $(\bar{\mathbf{x}}_t, \bar{\mathbf{x}}_{t-1}) \in \mathcal{N}^r$ with $\bar{\mathbf{x}}_t \neq \mathbf{x}_t$, $\bar{\mathbf{x}}_{t-1} \neq \mathbf{x}_{t-1}$:*

i. $0 < k(\mathbf{x}_t, \bar{\mathbf{x}}_t, \mathbf{x}_{t-1}, \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t) < C < \infty$ *for any $\mathbf{c}_t \in \mathcal{C}_t$ and some constant C;*

ii. $k(\mathbf{x}_t, \bar{\mathbf{x}}_t, \mathbf{x}_{t-1}, \bar{\mathbf{x}}_{t-1}, \bar{\mathbf{c}}_t) \neq k(\mathbf{x}_t, \bar{\mathbf{x}}_t, \mathbf{x}_{t-1}, \bar{\mathbf{x}}_{t-1}, \tilde{\mathbf{c}}_t)$*, where*

$$k(\mathbf{x}_t, \bar{\mathbf{x}}_t, \mathbf{x}_{t-1}, \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t) = \frac{p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t) p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t}(\bar{\mathbf{x}}_t \mid \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t)}{p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t}(\bar{\mathbf{x}}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t) p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t}(\mathbf{x}_t \mid \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t)}. \tag{1}$$

Assumption justification. Conceptually, Assumption 3 requires that k, which captures second-order variations in transition dynamics at time $t-1$ and t under the latent variable $\mathbf{c}$, yields distinct values for different $\mathbf{c}$'s. This requirement is typically met in RL, as varied latent dynamics or rewards often cause significant, observable shifts in behavior. *Crucially, this variability is precisely what motivates the need for the identification of the latent variable $\mathbf{c}_t$, as it governs meaningful differences in learning underlying decision-making process.*

Under these assumptions, we establish an identifiability theory that characterizes the conditions under which the latent factors can be recovered, and specifies the level of identifiability that can be achieved.

Theorem 1 (Identifiability on Posterior Distribution)**.** *Under Assumptions 1-3, the posterior distribution of latent factor with consecutive observations $p(\mathbf{c}_t \mid \mathbf{x}_{t-2:t+1})$ can be identifiable up to an invertible transformation on the latents $\hat{\mathbf{c}}_t = h(\mathbf{c}_t)$, where $\hat{\mathbf{c}}_t$ is estimated latents and h is an invertible function.*

The proof is in Appendix B.2. Theorem 1 indicates that a short temporal window of observations contains sufficient information to *recover the posterior distribution over the true latent factors* (up to an invertible transformation) in an online manner, without requiring access to the full trajectory.

4 Latent-Aware Adaptive Diffusion Planner and Policy

Building on Theorem 1, we introduce the `Ada-Diffuser` framework for learning and planning with latent identification. As illustrated in Fig. 2, `Ada-Diffuser` models the trajectory generation process via two modules: (1) **latent factor identification block**, which estimates the sequence of latent variables from the observable trajectories; and (2) **causal diffusion model**, which learns the underlying generative process of the RL trajectories. The learned model can then be used for both planning and policy learning by generating future trajectories or actions conditioned on the historical information, together with the inferred latent structure and task-specific demands.

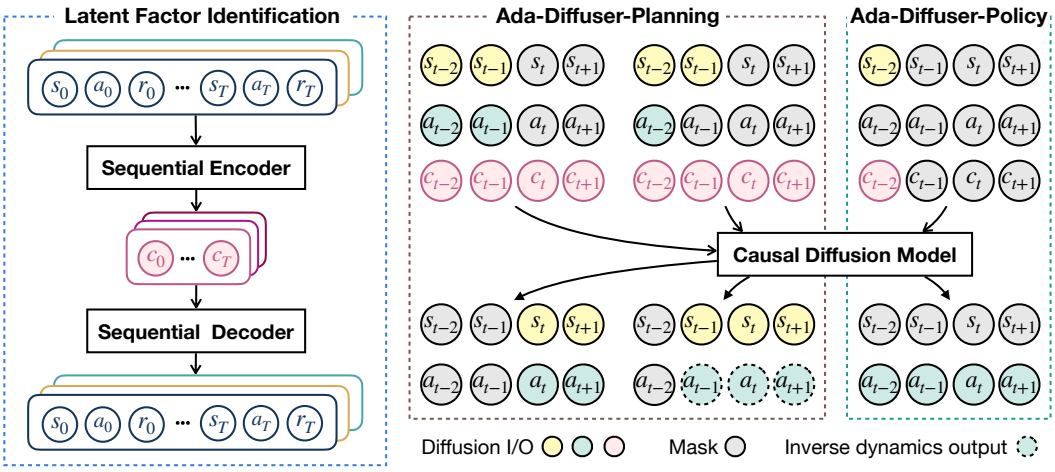

Figure 2: Overview of the `Ada-Diffuser` framework. The modular design consists of two main stages: latent context identification (Stage 1, Section 4.2), followed by a causal diffusion model (Stage 2, Section 4.3) that models the generative structure of the trajectories. The learned model is then used for planning or policy learning conditioned on the inferred latent context.

In this section, we first present a general formulation of conditional diffusion modeling with latent variables. We then describe the two modules of `Ada-Diffuser` in detail (Fig. 2). The complete algorithmic pseudocode of the training and inference procedures are given in Appendix D.1.

4.1 Latent-Augmented Diffusion Model for Planning and Policy Learning

Without loss of generality, we denote the observable trajectory as $\boldsymbol{\tau}_x$, which may correspond to a state-action sequence $\boldsymbol{\tau}_{sa}$ or a state-only sequence $\boldsymbol{\tau}_s$, depending on the task setting. To incorporate latent structure, we augment the observable trajectory with the estimated latent context, yielding the full trajectory representation $\boldsymbol{\tau} = [\boldsymbol{\tau}_x, \boldsymbol{\tau}_c]$, where $\boldsymbol{\tau}_c$ denotes the inferred sequence of latent variables.

We train a conditional diffusion model [58, 59] to generate trajectories conditioned on desired attributes $\boldsymbol{y}(\boldsymbol{\tau})$ (e.g., reward or goal specification) and the identified $\mathbf{c}$. The denoising model ϵ_θ is trained to predict the noise added during the forward diffusion process via the objective: $\mathcal{L}_{\text{diff}} = \mathbb{E}_{\boldsymbol{\tau}^0, \boldsymbol{y}, t, \boldsymbol{\epsilon}} \left[\| \epsilon_\theta(\boldsymbol{\tau}^t, t, \boldsymbol{y}(\boldsymbol{\tau}), \mathbf{c}) - \boldsymbol{\epsilon} \|^2 \right]$, where $\boldsymbol{\tau}^0$ is a clean trajectory sample, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, and the noisy trajectory at diffusion step t is constructed as: $\boldsymbol{\tau}^t = \sqrt{\bar{\alpha}_t}\boldsymbol{\tau}^0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, where $\bar{\alpha}_t$ denotes the cumulative product of the forward noise schedule. Here, the superscript t indexes diffusion steps, and should not be confused with the environment time step indices within the trajectory.

`Ada-Diffuser` can flexibly adapt to generate different components of the trajectory depending on the task. In the *planning* setting, the model generates full trajectories $\boldsymbol{\tau} = \{\mathbf{x}_t, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+T_p}\}$, where T_p denotes the planning horizon. Here, $\mathbf{x}_t$ may have two cases: (i) $\mathbf{x}_t = \{\mathbf{s}_t, \mathbf{a}_t\}$, when both states and actions are generated, (ii) $\mathbf{x}_t = \{\mathbf{s}_t\}$, when only states are generated. In the latter case, we train an inverse dynamics model (IDM) [70] to infer the corresponding actions from state transitions. In the *policy learning* setting, the model generates only actions, i.e., $\boldsymbol{\tau} = \{\mathbf{a}_{t+1}, \mathbf{a}_{t+2}, \ldots, \mathbf{a}_{t+T_a}\}$, where T_a is the action generation horizon. While multi-step action generation methods (e.g., DP [7]) can also be viewed as a form of planning [71], for generality, we categorize such settings under the policy framework. `Ada-Diffuser-Policy` accommodates both variants: multi-step action generation ($T_a > 1$), as in DP, and single-step decision-making ($T_a = 1$), as in IDQL [10].

4.2 Stage 1: Offline Latent Factor Identification

Based on Theorem 1, we structure the latent inference process around *temporal blocks*, using short segments of trajectories to identify the latent context at each time step. We adopt a variational inference framework [72] in which the latent variable $\mathbf{c}_t$ is inferred block-wise. The prior distribution is conditioned on the latent variable from the previous step and the in-block history, while the posterior additionally incorporates future observations. Specifically, given a trajectory block $t - T_x : t + 1$, where T_x is the block size, we have prior $p_\phi(\mathbf{c}_t \mid \mathbf{c}_{t-1})$, and posterior $q_\psi(\mathbf{c}_t \mid \mathbf{x}_{t-T_x:t+1})$, where $\mathbf{x}$

denotes the observed variables and may correspond to $\{\mathbf{s}\}$, $\{\mathbf{s}, \mathbf{a}\}$, or $\{\mathbf{s}, \mathbf{a}, r\}$. We then optimize the evidence lower bound (ELBO) of the observed trajectories:

$$\mathcal{L}_{\text{ELBO},t} = \mathbb{E}_{q_\psi(\mathbf{c}_t|\mathbf{x}_{t-T_x:t+1})}\left[-\log p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t)\right] + D_{\text{KL}}\left(q_\psi(\mathbf{c}_t \mid \mathbf{x}_{t-T_x:t+1}) \,\|\, p_\phi(\mathbf{c}_t \mid \mathbf{c}_{t-1})\right).$$

Here, the reconstruction term, $-\log p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t)$ is instantiated based on the available observation modalities. Specifically, (i) when only states are observed, the model reconstructs $\mathbf{s}_t$ conditioned on $(\mathbf{s}_{t-1}, \mathbf{c}_t)$; and (ii) when rewards are available, the model also reconstructs r_t from $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c}_t)$. The stage is learned through a sequential encoder and decoder (l.h.s., Fig. 2).

4.3 Stage 2: Causal Diffusion Model

We propose a **causal diffusion model** for learning the generative process described in Sec.3.1. By "causal," we refer to the modeling of the true underlying data generation process, which incorporates two key desiderata: (1) the *autoregressive process* inherent in temporal sequential RL trajectories; and (2) the *latent factor process*, capturing the causal influence of the unobserved context variables $\mathbf{c}_t$ on the observations (e.g., $\mathbf{x}_t = [\mathbf{s}_t, \mathbf{a}_t, r_t]$). Thus, different from other diffusion-based planners or policies (Sec.2), we have the following designs.

Autoregressive Denoising To model the autoregressive structure of trajectory generation, and following the recent advances in autoregressive diffusion [48, 49, 57], we introduce a **causal denoising schedule**. Under this mechanism, each time step within a local temporal block is assigned a denoising schedule that depends both on its temporal distance from the conditioning anchor and on the inferred latent variables. This reflects the intuition that later time steps exhibit higher uncertainty. Specifically, for a trajectory of length T, we assign monotonically increasing noise levels $\{k_1, \dots, k_T\}$, sampled linearly as $k_i = \frac{i}{T}K$ where $i \in \{1, \dots, T\}$ and K denotes the maximum diffusion step.

Given the inferred latent context $\hat{\mathbf{c}}_{0:T}$, the model performs autoregressive denoising over the block in T steps. The overall denoising process is defined as:

$$p_\theta\left(\mathbf{x}_0^0, \dots, \mathbf{x}_{T-1}^0 \mid \mathbf{x}_0^{k_1}, \dots, \mathbf{x}_{T-1}^{k_T}, \hat{\mathbf{c}}_{0:T}\right), \tag{2}$$

where $\mathbf{x}_i^{k_i}$ denotes the noisy observation at time step i, and $\mathbf{x}_i^0$ is the clean, denoised output.

Specifically, the first denoising step is: $p_\theta(\mathbf{x}_0^0, \mathbf{x}_1^{k_1}, \dots, \mathbf{x}_{T-1}^{k_{T-1}} \mid \mathbf{x}_0^{k_1}, \dots, \mathbf{x}_{T-1}^{k_T}, \hat{\mathbf{c}}_{0:T})$, where the first observation $\mathbf{x}_0$ has been fully denoised and other observations are partially denoised, followed by the second step: $p_\theta(\mathbf{x}_1^0, \mathbf{x}_2^{k_1}, \dots, \mathbf{x}_{T-1}^{k_{T-2}} \mid \mathbf{x}_0^0, \mathbf{x}_1^{k_1}, \dots, \mathbf{x}_{T-1}^{k_{T-1}}, \hat{\mathbf{c}}_{0:T})$, and finally until all observations are denoised: $p_\theta(\mathbf{x}_{T-1}^0 \mid \mathbf{x}_0^0, \dots, \mathbf{x}_{T-2}^0, \mathbf{x}_{T-1}^{k_1}, \hat{\mathbf{c}}_{0:T})$.

Denoise-and-refine Mechanism To obtain the latent context estimates $\hat{\mathbf{c}}_{0:T}$, one option is to use the pre-trained prior network p_ϕ from Stage 1. Theorem 1 indicates that both historical and future observations are required for recovering the latents. However, these future observations are not accessible during online inference, which results in a mismatch between identifiability requirements and available information. To address this challenge while preserving the causal structure of the generative process, we propose a *denoise-and-refine mechanism* that alternates between denoising the observable sequences and refining the latent estimates, and is applied consistently during both training and inference to ensure high-quality latent context modeling under partial observability.

Training: Given a noisy input $\mathbf{x}_t^{k_t}$ with noise level k_t, we first sample an initial latent context from the prior: $\hat{\mathbf{c}}_t^{\text{prior}} \sim p_\phi(\mathbf{c}_t \mid \mathbf{c}_{t-1})$, and use it to denoise the observation: $\hat{\mathbf{x}}_t^{(0)} = \epsilon_\theta(\mathbf{x}_t^{k_t}, k_t, \hat{\mathbf{c}}_t^{\text{prior}})$. Then we infer the latent using the posterior network, conditioned on a broader temporal window including future observations (accessible in offline data): $\hat{\mathbf{c}}_t^{\text{post}} \sim q_\psi(\mathbf{c}_t \mid \mathbf{x}_{t-k:t+1})$, and obtain a refined denoised prediction: $\hat{\mathbf{x}}_t^{(0)\prime} = \epsilon_\theta(\mathbf{x}_t^{k_t}, k_t, \hat{\mathbf{c}}_t^{\text{post}})$.

We have two reconstruction losses: one from the prior-sampled latent, $\mathcal{L}_{\text{prior}} = \|\hat{\mathbf{x}}_t^{(0)} - \mathbf{x}_t^0\|^2$, and one from the posterior-sampled latent, $\mathcal{L}_{\text{post}} = \|\hat{\mathbf{x}}_t^{(0)\prime} - \mathbf{x}_t^0\|^2$. To encourage the posterior latent to produce better reconstructions, we introduce a contrastive improvement loss: $\mathcal{L}_{\text{contrast}} = \max\{0, \mathcal{L}_{\text{prior}} - \mathcal{L}_{\text{post}}\}$. The final objective for this denoise-and-refine step is: $\mathcal{L}_{\text{d-r}} = \mathcal{L}_{\text{post}} + \lambda_{\text{prior}}\mathcal{L}_{\text{prior}} + \lambda_{\text{contrast}}\mathcal{L}_{\text{contrast}}$, where λ_{prior} and $\lambda_{\text{contrast}}$ are weighting coefficients.

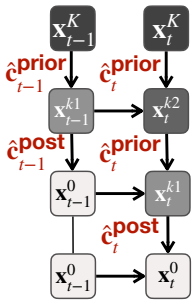

Figure 3: A 2-step example of the zig-zag sampling.

Inference: During inference, future observations are not available, which prevents direct use of the posterior network for latent inference. To address this, we adopt a *zig-zag sampling strategy* that

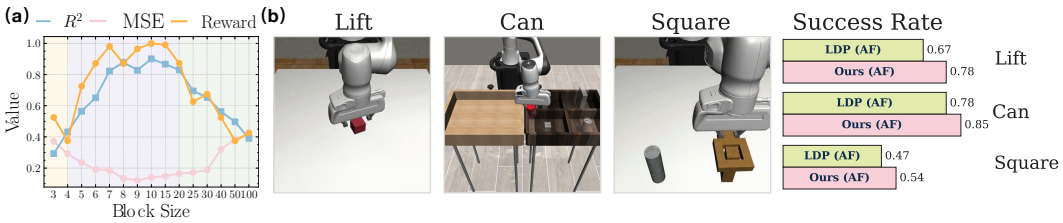

Figure 4: (a). Identification Results (i.e., Linear Probing MSE, R^2) and normalized rewards on the Cheetah environment with time-varying wind as the latent factor, evaluated across different block sizes. (b). Results (i.e., average success rate) on planning with action-free demonstrations on Robomimic benchmark. "AF" denotes Action-free.

combines autoregressive denoising with latent refinement. Specifically, we first sample the entire trajectory by applying the forward diffusion process with the maximum noise level K. We then perform autoregressive denoising across time.

For each time step t, we begin by denoising $\mathbf{x}_t^K$ to an intermediate noise level k_1 using $\hat{\mathbf{c}}_t$ sampled from the prior: $\hat{\mathbf{c}}_t^{\text{prior}} \sim p_\phi(\mathbf{c}_t \mid \mathbf{c}_{t-1})$. We then obtain updated $\hat{\mathbf{c}}_t$ from the posterior latent distribution $\hat{\mathbf{c}}_t^{\text{post}} \sim q_\psi(\mathbf{c}_t \mid \mathbf{x}_{t-k:t-1}^0, \mathbf{x}_t^{k_1}, \mathbf{x}_{t+1}^{k_2})$, which is conditioned on the denoised history, the intermediate step with noise level k_1, and the next step with noise level k_2. We then use $\hat{\mathbf{c}}_t^{\text{post}}$ as the input to further denoise $\mathbf{x}_t^{k_1}$ to $\mathbf{x}_t^0$. An illustration of the zig-zag inference process is provided in Fig. 3[1].

In summary, `Ada-Diffuser` provides a unified framework for planning and policy learning, which jointly models both observational and latent data components. It leverages autoregressive noise scheduling to reflect temporal structure, integrates latent context identification by the denoise-and-refine mechanism, and employs zig-zag sampling for online latent inference and state-action generation. This framework accommodates a wide range of scenarios, including latent dynamics/rewards, learning from action-free data with latent actions, and both state- and image-based environments. All variants share the same core, with task-specific modifications to the model components and diffusion input/output (I/O). Details of these architectural and I/O variations are in Appendix H.

5 Experiments

In this section, we provide the empirical evaluation to answer the following questions: (1) *Latent Identification*: How well can `Ada-Diffuser` capture latent factors in the environment? (2) *Learning with Latent Factors*: How effective is `Ada-Diffuser` in planning and control when learning with the latent context on dynamics and reward? And can `Ada-Diffuser` infer latent actions from action-free demonstrations? (3) *Learning with Environments w/o Explicit Latents*: In environments without explicit latent factors, can modeling latent processes still bring performance gains? (4) *Ablation Studies*: What is the impact of key design choices in the framework?

Benchmarks We consider a diverse set of benchmarks, including Mujoco-based locomotion tasks (`Cheetah`, `Ant`, `Walker`), a robot navigation task (`Maze2D`), and a robot arm control task (`Franka-Kitchen`) [73], all from the D4RL benchmark suite [74]. We also consider robotic manipulation tasks from `RobotMimic` [75] and `LIBERO-10` [76]. A detailed description and illustration of these environments is provided in Appendix F. We introduce latent factors affecting both dynamics ($\mathbf{c}^s$) and reward functions ($\mathbf{c}^r$) in the Cheetah and Ant environments, considering two types of variations: episodic changes (E) and fine-grained, time-varying step-wise changes (S). The specific change functions for each setting are detailed in Appendix F.1. For evaluating latent action modeling, we follow the setup from LDP [77], using action-free, pixel-based demonstrations from the LIBERO benchmark [76].

Baselines We compare `Ada-Diffuser` with a diverse set of baselines for fair and comprehensive evaluation. *(1) Vanilla diffusion models*: For planning, we consider Diffuser [5] and DD [6]. For policy learning, we include DP and IDQL [10]. We also evaluate LDCQ [78], which learns a latent skill space and optimizes a value function conditioned on both states and latent skills. *(2) Latent context modeling*: We include MetaDiffuser [53] that learns contextual representations from multiple environments. We also consider using LILAC [29] and DynaMITE [45] which models

[1]A larger illustration with 4 steps are given in Appendix Fig. A2.

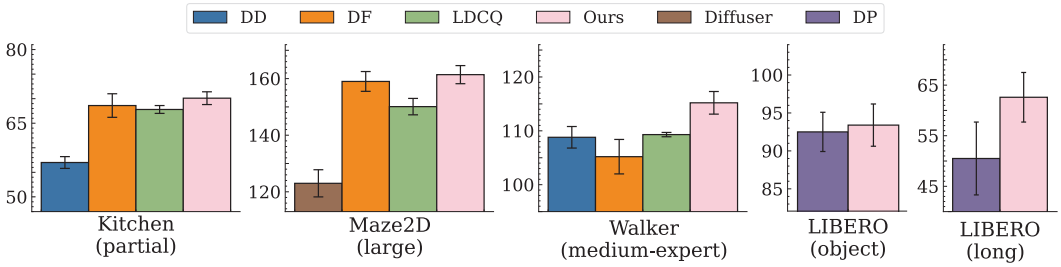

Figure 5: Results comparing `Ada-Diffuser` with baselines in environments without explicitly designed latent factors. Complete results are provided in Appendix Table A5–A8. Our diffusion planner/policy is based on the design choices of DD (Kitchen, Maze2D, Walker) or DP (LIBERO).

Environment	Diffuser	DF	MetaDiffuser	Diffuser + DynaMITE	Diffuser + LILAC	Ours
Cheetah-Wind-E ($\mathbf{c}^s$)	-120.4 ± 12.7	-105.8 ± 9.6	-89.7 ± 6.5	-79.2 ± 11.0	-95.3± 7.4	**-68.9**± 7.6
Cheetah-Wind-S ($\mathbf{c}^s$)	-148.5± 9.8	-102.0± 10.2	-106.8± 11.4	-94.3± 9.6	-105.6± 14.5	**-73.5**± 8.7
Cheetah-Vel-E ($\mathbf{c}^r$)	-102.4± 18.2	-85.6± 18.3	-69.2± 7.5	-76.3± 11.7	-62.6± 11.1	**-45.8**± 9.5
Ant-Dir-E ($\mathbf{c}^r$)	188.6± 39.2	195.4± 47.0	245.9± 41.0	262.8± 27.5	229.4± 32.6	**285.3**± 24.5

Table 1: Results on `Ada-Diffuser-Planner` with latent factors that affects dynamics and rewards. $\mathbf{c}^s$ and $\mathbf{c}^s$ indicate the changes on dynamics and reward, E and S represent the episodic and time-step changes. All results are averaged over 5 random seeds.

nonstationarity in RL through latent context learning using belief states. For a fair comparison, we integrate their context modules into diffusion planners and policies as plug-in components (detailed analysis in Appendix H.1). *(3) Latent action modeling*: We compare with LDP [77] with action-free demonstrations for planning.

Architecture Choices For latent factor identification, we use GRU [79] embedding with MLP layers as both prior and posterior encoders to produce Gaussian distribution over latents. For decoders, we use MLP layers. For planning and policy learning, we use UNet [80] or Transformers [81] as denoising networks and use MLPs to learn the IDM. For image-based settings, we use VAE [72] pretrained on demonstrations. All detailed hyperparameters are given in Appendix D.2.

Results on Latent Identification To verify our identification theory, we evaluate model performance under different block sizes that contain varying amounts of temporal context. We include settings where all blocks have sufficient observations, as well as a challenging case with insufficient observations (i.e., without access to future observations). To quantify the quality of the learned latent representations, we adopt linear probing and the coefficient of determination R^2 as the evaluation metric. The results, together with normalized results, are shown in Fig. 4(a). The yellow region indicates settings with insufficient observations, resulting in lower identification results. The purple region corresponds to sufficient observations and yields relatively strong performance, and the green region reflects larger block sizes, which lead to degraded results due to redundant information or inherent difficulty for optimization. Notably, the reward is positively associated with the accuracy of latent identification, validating the importance of identifying latent factors in RL trajectories.

Results on Decision-making *Group I: Latent factors on dynamics and reward:* Table 1-2 presents the results of planning and policy learning under latent factors that affect dynamics and rewards in locomotion tasks. Additional results, including oracle variants and meta-learned versions of `Ada-Diffuser` that use ground-truth latents as input, are provided in Appendix Table A3–A4. From the results, we observe that `Ada-Diffuser` consistently achieves the best performance, with a significant margin over all baselines. In particular, it outperforms Diffusion planners and policies even when those models are enhanced with latent context modules such as DynaMITE and LILAC, which are most comparable to our setting. Furthermore, `Ada-Diffuser` outperforms DF, highlighting the effectiveness of our causal diffusion model coupled with latent context identification.

Group II: Latent Actions: Following Xie et al. [77], we consider learning from action-free demonstration data, where actions are treated as latent factors to be inferred. We adopt the same setup as in [77], using a pre-trained visual encoder obtained via a VAE to learn the latent space from pixel observations.

Environment	DP	DP + DynaMITE	Ours + DP	IDQL	IDQL + DynaMITE	Ours + IDQL
Cheetah-Wind-E ($\mathbf{c}^s$)	-104.8 ± 10.9	-72.2± 5.9	**-58.5**± 4.6	-97.5± 9.4	-59.0± 11.2	**-48.5**± 7.9
Cheetah-Wind-S ($\mathbf{c}^s$)	-120.6± 11.5	-76.5± 15.6	**-52.9**± 9.8	-87.8± 12.2	-63.4± 6.7	**-48.0**± 7.2
Cheetah-Vel-E ($\mathbf{c}^r$)	-87.9± 6.5	-72.7± 5.8	**-41.0**± 7.2	-80.2± 11.4	-59.4± 6.5	**-38.6**± 7.7
Ant-Dir-E ($\mathbf{c}^r$)	182.5± 41.2	275.2± 27.0	**290.4**± 49.4	204.6± 25.6	269.3± 29.5	**295.8**± 32.7

Table 2: Results on `Ada-Diffuser-Policy` with latent factors. All results are averaged over 5 random seeds.

We then train a latent planner and an IDM using a diffusion-based approach. Unlike prior work, our diffusion-based latent planner additionally incorporates latent factors $\mathbf{c}$ to model latent context. Importantly, we train only the planner using additional action-free demonstrations. Detailed training procedures are provided in Appendix G.1. Results on several tasks in `Robomimic` benchmark show that we can bring improvements on all tasks via modeling the latent process supplementary to the latent planner in [77]. Here, the IDM is trained solely on expert demonstrations. Complete results are provided in Appendix Table A2.

Group III: Environments w/o Explicitly Designed Latents: Crucially, in this scenario, the latent variable $\mathbf{c}$ effectively serves as a form of Bayesian filtering over the observed trajectories, capturing the inherent stochasticity in the data (a more detailed discussion in Appendix D.3). Such variability commonly arises from system noise, expert action noise, or high-level unobserved factors. The results, shown in Fig. 5 (full results provided in Appendix Table A5–A8), support this interpretation. Even in environments without explicitly designed latent contexts, incorporating latent modeling allows `Ada-Diffuser` to achieve performance that is comparable to or better than these baselines. These findings suggest that our framework can consistently capture implicit latent process in the data, improving both trajectory modeling and downstream planning.

Ablation Studies We conduct ablation studies to evaluate the contributions of key components in our framework. For *latent factor identification*, Fig. 4(a) shows the effect of different temporal block sizes, illustrating the benefit of incorporating future observations during inference. For the *causal diffusion model*, we examine the impact of the following design choices: (i) removing the refinement step using posterior samples of $\hat{\mathbf{c}}$ during training (*w/o refine*); (ii) removing zig-zag sampling and relying solely on $\hat{\mathbf{c}}$ (*w/o zig-zag*); (iii) replacing the causal noise schedule with a fixed noise level across time steps (vanilla diffusion) or with random noise scaling as in DF [48] (*same NS, random NS*). The results in Table 3 demonstrate the effectiveness of these modules in our framework in both settings: with and without explicit latent factors. Additional ablations are provided in Appendix I.1, including further evaluations on each module and training/inference time.

Cases	Cheetah ($\mathbf{c}^s$)	LIBERO
Original	**-73.5**	**93.4**
w/o refine	-82.0	90.2
w/o zig-zag	-91.6	91.6
same NS	-89.7	85.2
random NS	-84.6	88.5

Table 3: Ablation on Design Choices on Cheetah-Wind-S (planner) and LIBERO (DP-policy).

6 Conclusions

In this work, we demonstrate that identifying latent factors from sequential observations is critical for effective decision-making. We provide theoretical results that establish conditions under which latent variables can be identified using small temporal blocks of observations. This insight enables a principled integration of latent identification into a diffusion-based generative framework, allowing us to capture the underlying causal process while maintaining scalability. Our proposed `Ada-Diffuser` is broadly applicable to a variety of settings, including planning and control tasks with or without explicit latent structure, and even action-free demonstrations. Empirical results across diverse benchmarks show substantial improvements, validating the effectiveness of our method not only in environments with designed latent factors but also in general settings where latent structure is implicit but influential.

Limitations and future work One current limitation of our approach is its relatively slow inference time, due to the iterative nature of the sampling process. This may be mitigated by applying acceleration techniques such as parallel sampling [82]. In future work, we aim to extend our framework to other decision-making foundation models, including vision-language-action (VLA) models [14–22].

References

[1] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[2] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[3] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international conference on machine learning*, pages 27042–27059. PMLR, 2022.

[4] Deqian Kong, Dehong Xu, Minglu Zhao, Bo Pang, Jianwen Xie, Andrew Lizarraga, Yuhao Huang, Sirui Xie, and Ying Nian Wu. Latent plan transformer for trajectory abstraction: Planning as latent space inference. *Advances in Neural Information Processing Systems*, 37: 123379–123401, 2024.

[5] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[6] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[7] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[8] Allen Z. Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mEpqHvbD2h.

[9] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

[10] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies, 2023.

[11] Yuhui Chen, Haoran Li, and Dongbin Zhao. Boosting continuous control with consistency policy. *arXiv preprint arXiv:2310.06343*, 2023.

[12] Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pages 22825–22855. PMLR, 2023.

[13] Zihan Ding, Amy Zhang, Yuandong Tian, and Qinqing Zheng. Diffusion world model: Future modeling beyond step-by-step rollout for offline reinforcement learning. *arXiv preprint arXiv:2402.03570*, 2024.

[14] Yilun Du, Mengjiao Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B Tenenbaum, et al. Video language planning. *arXiv preprint arXiv:2310.10625*, 2023.

[15] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

[16] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.

[17] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[18] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[19] Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.

[20] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

[21] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[22] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi\_\{0.5\}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[23] Mikko Lauri, David Hsu, and Joni Pajarinen. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, 2022.

[24] Zhiyu Huang, Chen Tang, Chen Lv, Masayoshi Tomizuka, and Wei Zhan. Learning online belief prediction for efficient pomdp planning in autonomous driving. *IEEE Robotics and Automation Letters*, 2024.

[25] Milos Hauskrecht and Hamish Fraser. Planning treatment of ischemic heart disease with partially observable markov decision processes. *Artificial intelligence in medicine*, 18(3): 221–244, 2000.

[26] Daniel E Ehrmann, Shalmali Joshi, Sebastian D Goodfellow, Mjaye L Mazwi, and Danny Eytan. Making machine learning matter to clinicians: model actionability in medical decision-making. *NPJ Digital Medicine*, 6(1):7, 2023.

[27] Gianluca Brero, Alon Eden, Darshan Chakrabarti, Matthias Gerstgrasser, Amy Greenwald, Vincent Li, and David C Parkes. Stackelberg pomdp: A reinforcement learning approach for economic design. *arXiv preprint arXiv:2210.03852*, 2022.

[28] Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: Variational bayes-adaptive deep rl via meta-learning. *Journal of Machine Learning Research*, 22(289):1–39, 2021.

[29] Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst continual structured non-stationarity. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11393–11403. PMLR, 18–24 Jul 2021.

[30] Gokul Swamy, Sanjiban Choudhury, J Bagnell, and Steven Z Wu. Sequence model imitation learning with unobserved contexts. *Advances in Neural Information Processing Systems*, 35: 17665–17676, 2022.

[31] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. *Advances in neural information processing systems*, 36:80375–80395, 2023.

[32] Annie Xie, Lisa Lee, Ted Xiao, and Chelsea Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3153–3160. IEEE, 2024.

[33] Joey Hejna, Chethan Bhateja, Yichen Jiang, Karl Pertsch, and Dorsa Sadigh. Re-mix: Optimizing data mixtures for large scale imitation learning. *arXiv preprint arXiv:2408.14037*, 2024.

[34] Jensen Gao, Annie Xie, Ted Xiao, Chelsea Finn, and Dorsa Sadigh. Efficient data collection for robotic manipulation via compositional generalization. *arXiv preprint arXiv:2403.05110*, 2024.

[35] Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *Advances in Neural Information Processing Systems*, 37:124420–124450, 2024.

[36] Tanmay Gangwani, Joel Lehman, Qiang Liu, and Jian Peng. Learning belief representations for imitation learning in pomdps. In *uncertainty in artificial intelligence*, pages 1061–1071. PMLR, 2020.

[37] Milos Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of artificial intelligence research*, 13:33–94, 2000.

[38] Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo A Pires, and Rémi Munos. Neural predictive belief representations. *arXiv preprint arXiv:1811.06407*, 2018.

[39] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *International conference on machine learning*, pages 2117–2126. PMLR, 2018.

[40] Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. Temporal difference variational auto-encoder. *arXiv preprint arXiv:1806.03107*, 2018.

[41] Arun Venkatraman, Nicholas Rhinehart, Wen Sun, Lerrel Pinto, Martial Hebert, Byron Boots, Kris Kitani, and J Bagnell. Predictive-state decoders: Encoding the future into recurrent networks. *Advances in Neural Information Processing Systems*, 30, 2017.

[42] Nathanael Bosch, Jan Achterhold, Laura Leal-Taixé, and Jörg Stückler. Planning from images with deep latent gaussian process dynamics. In *Learning for Dynamics and Control*, pages 640–650. PMLR, 2020.

[43] Cuong C Nguyen, Thanh-Toan Do, and Gustavo Carneiro. Probabilistic task modelling for meta-learning. In *Uncertainty in Artificial Intelligence*, pages 781–791. PMLR, 2021.

[44] Andrew Wang, Andrew C Li, Toryn Q Klassen, Rodrigo Toro Icarte, and Sheila A McIlraith. Learning belief representations for partially observable deep rl. In *International Conference on Machine Learning*, pages 35970–35988. PMLR, 2023.

[45] Anthony Liang, Guy Tennenholtz, Chih-wei Hsu, Yinlam Chow, Erdem Bıyık, and Craig Boutilier. Dynamite-rl: A dynamic model for improved temporal meta-reinforcement learning. *arXiv preprint arXiv:2402.15957*, 2024.

[46] Minseung Lee, Hyeonseo Cho, and Sungjin Ahn. Pomdiffuser: Long-memory meets long-planning for pomdps.

[47] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[48] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

[49] Desai Xie, Zhan Xu, Yicong Hong, Hao Tan, Difan Liu, Feng Liu, Arie Kaufman, and Yang Zhou. Progressive autoregressive video diffusion models. *arXiv preprint arXiv:2410.08151*, 2024.

[50] Sand-AI. Magi-1: Autoregressive video generation at scale, 2025. URL https://static.magi.world/static/files/MAGI_1.pdf.

[51] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=mLcmdlEUxy-.

[52] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

[53] Fei Ni, Jianye Hao, Yao Mu, Yifu Yuan, Yan Zheng, Bin Wang, and Zhixuan Liang. Metadiffuser: Diffusion model as conditional planner for offline meta-rl. In *International Conference on Machine Learning*, pages 26087–26105. PMLR, 2023.

[54] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

[55] Kaifeng Gao, Jiaxin Shi, Hanwang Zhang, Chunping Wang, and Jun Xiao. Vid-gpt: Introducing gpt-style autoregressive generation in video diffusion models. *arXiv preprint arXiv:2406.10981*, 2024.

[56] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[57] Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Jian Jiao, Juntao Li, Jian Guo, Nan Duan, and Weizhu Chen. Ar-diffusion: Auto-regressive diffusion model for text generation. *Advances in Neural Information Processing Systems*, 36:39957–39974, 2023.

[58] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[59] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[60] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[61] Judea Pearl. Causal inference. *Causality: objectives and assessment*, pages 39–58, 2010.

[62] Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *IJCAI: proceedings of the conference*, volume 2016, page 1432, 2016.

[63] Christian Perez, Felipe Petroski Such, and Theofanis Karaletsos. Generalized hidden parameter mdps: Transferable model-based rl in a handful of trials. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5403–5411, 2020.

[64] James John Martin. *Some Bayesian decision problems in a Markov chain.* PhD thesis, Massachusetts Institute of Technology, 1965.

[65] Michael O'Gordon Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. University of Massachusetts Amherst, 2002.

[66] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[67] Nelson Dunford and Jacob T. Schwartz. *Linear Operators*. John Wiley & Sons, New York, 1971.

[68] Yingyao Hu and Susanne M Schennach. Instrumental variable treatment of nonclassical measurement error models. *Econometrica*, 76(1):195–216, 2008.

[69] Yingyao Hu and Matthew Shum. Nonparametric identification of dynamic models with unobserved state variables. *Journal of Econometrics*, 171(1):32–44, 2012.

[70] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sP1fo2K9DFG.

[71] Zhengbang Zhu, Hanye Zhao, Haoran He, Yichao Zhong, Shenyu Zhang, Haoquan Guo, Tingting Chen, and Weinan Zhang. Diffusion models for reinforcement learning: A survey. *arXiv preprint arXiv:2311.01223*, 2023.

[72] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *The International Conference on Learning Representations (ICLR)*, 2014.

[73] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Proceedings of the Conference on Robot Learning (CoRL)*. PMLR, 2020.

[74] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[75] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[76] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

[77] Amber Xie, Oleh Rybkin, Dorsa Sadigh, and Chelsea Finn. Latent diffusion planning for imitation learning. *International Conference on Machine Learning (ICML)*, 2025.

[78] Siddarth Venkatraman, Shivesh Khaitan, Ravi Tej Akella, John Dolan, Jeff Schneider, and Glen Berseth. Reasoning with latent diffusion in offline reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tGQirjzddO.

[79] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[80] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[81] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[82] Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:4263–4276, 2023.

[83] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

[84] Raymond J Carroll, Xiaohong Chen, and Yingyao Hu. Identification and estimation of nonlinear models using two samples with nonclassical measurement errors. *Journal of nonparametric statistics*, 22(4):379–399, 2010.

[85] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

[86] Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International Conference on Machine Learning*, pages 9767–9779. PMLR, 2021.

[87] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[88] Zhe Chen et al. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182(1):1–69, 2003.

[89] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[90] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[91] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[92] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[93] Wenhao Li. Efficient planning with latent diffusion. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=btpgDo4u4j.

[94] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. *arXiv preprint arXiv:2302.01877*, 2023.

[95] Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in neural information processing systems*, 36:64896–64917, 2023.

[96] Wei Xiao, Tsun-Hsuan Wang, Chuang Gan, Ramin Hasani, Mathias Lechner, and Daniela Rus. Safediffuser: Safe planning with diffusion probabilistic models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ig2wk7kK9J.

[97] Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, Dragomir Anguelov, et al. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9644–9653, 2023.

[98] Anurag Ajay, Seungwook Han, Yilun Du, Shaung Li, Abhi Gupta, Tommi Jaakkola, Josh Tenenbaum, Leslie Kaelbling, Akash Srivastava, and Pulkit Agrawal. Compositional foundation models for hierarchical planning. *arXiv preprint arXiv:2309.08587*, 2023.

[99] Zhixuan Liang, Yao Mu, Hengbo Ma, Masayoshi Tomizuka, Mingyu Ding, and Ping Luo. Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16467–16476, 2024.

[100] Zibin Dong, Yifu Yuan, Jianye HAO, Fei Ni, Yao Mu, YAN ZHENG, Yujing Hu, Tangjie Lv, Changjie Fan, and Zhipeng Hu. Aligndiff: Aligning diverse human preferences via behavior-customisable diffusion model. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=bxfKIYfHyx.

[101] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Appendix of Ada-Diffuser: Latent-Aware Adaptive Diffusion for Decision-Making

---

A Discussions and Overview

In this section, we expand on the design and motivation behind `Ada-Diffuser`, including the rationale for modeling latent factors in decision-making, key architectural choices, and additional analysis of the experimental results presented in Section 5. We then provide an overview of the remaining contents of this appendix.

A.1 Broader Impact

Our work aims to identify and leverage latent processes in generative decision-making, with applications in real-world domains such as robotics and healthcare. While these tasks may entail potential societal risks, we do not believe any specific concerns need to be highlighted here. Instead, by uncovering and modeling the underlying hidden processes, our approach promotes greater transparency in decision-making, which can ultimately lead to more reliable and trustworthy outcomes.

A.2 Discussions on the Core Idea

Q1: On Latent Modeling. *Why is it necessary to model latent processes when we already have access to a large amount of demonstration data?*

In many decision-making systems, there exist unobservable variables that influence both the dynamics and the reward structure. More generally, these latent variables often evolve over time. Such scenarios are common in real-world settings, for example, in robotic control, system dynamics can be affected by external forces (e.g., wind, friction), or by varying user demands (e.g., different target positions). In these cases, learning an optimal policy requires conditioning on the latent factors, especially when they are non-stationary or when transferring to new domains. Prior work has demonstrated the importance of latent variable modeling in both reinforcement learning (RL) and imitation learning (IL) [28, 45, 43, 52, 53, 29].

Even with access to large demonstration datasets, it remains difficult to ensure sufficient coverage over the full space of environmental or task-specific latent factors relevant to decision-making. This limitation has been widely acknowledged in recent efforts focused on analyzing data quality and designing data collection protocols to promote generalization [31–34]. However, most of these works target fixed or task-specific latent variables. In contrast, we consider a more general setting where latent factors evolve over time and are not predefined. Our framework provides theoretical guarantees for identifying such latent variables from partial observations and seamlessly integrates this identification process into diffusion models, enabling scalability across complex decision-making tasks.

Q2: On the Scenarios w/o Explicit Latents. *What does latent modeling represent when no explicit latent factors are defined, and why can it still benefit decision-making?*

Even in settings where all task-relevant observations are available, e.g., in locomotion tasks where full physical state information is provided, or in robotic manipulation with access to both proprioceptive and visual inputs, there may still exist underlying processes that are not directly observed. These

include domain-specific factors such as external forces (e.g., wind) or dynamically changing task goals (e.g., target positions), which can be viewed as implicit latent variables.

In the extreme case where such factors are also fully observed, latent modeling can still offer significant benefits. Specifically, it can capture residual stochasticity present in the environment or demonstration data, serving to explain variability not accounted for by observable features. The model can then identify meaningful structure from irrelevant or noisy variations, for instance, filtering out visual background artifacts that are not predictive of dynamics or optimal actions.

Moreover, since both the dynamics and optimal action distributions are typically stochastic rather than deterministic functions of the observed states, modeling latent processes helps approximate these stochastic mappings more effectively. By doing so, latent modeling supports improved representation learning, generative modeling, and planning, similar to Bayesian filtering, by structuring uncertainty and capturing influential variation in the decision-making process.

Q3: On the Identification Theory. *What does the identification theory establish, and how does it inform algorithm design?*

The identification theory (Theorem 1) establishes that the distribution over latent variables can be provably recovered from observable trajectories using only a small temporal window, specifically, a small temporal block of four time steps. This provides a general non-parametric theoretical guarantee that latent factors can be identified without requiring strong inductive biases or restrictive assumptions on the model class or functional form.

This "four-step" result has direct implications for algorithm design. It suggests that latent identification can be effectively performed using a short temporal block, which aligns naturally with block-wise generative modeling approaches such as diffusion models. These models operate over segments or chunks of data, and our theoretical results justify using local temporal blocks to infer latent variables in a principled and scalable manner.

A.3 Discussions on the Theoretical Assumptions and Results

Q4: On the Assumptions. *What do Assumption 2 (Distributional Variability) and Assumption 3 (Uniqueness of Spectral Decomposition) mean, and why are they considered mild?*

We expand on the intuition and practical relevance of these two assumptions below.

Distributional Variability (Assumption 2) refers to the requirement that the conditional distributions

$$p(\mathbf{x}_{t-2} \mid \mathbf{x}_{t+1}), \quad p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{c}_t), \quad \text{and} \quad p(\mathbf{x}_t \mid \mathbf{x}_{t-2}, \mathbf{x}_{t-1})$$

are sufficiently sensitive to variations in their input. That is, for different input pairs within a local neighborhood, the output distributions differ meaningfully, ensuring the system exhibits enough variability for identification. This assumption aligns with real-world decision-making settings (e.g., locomotion or robotic manipulation), where changes in inputs such as physical state, control policy, or reward function lead to observable changes in output distributions.

Uniqueness of Spectral Decomposition (Assumption 3) builds on this by ensuring that changes in the latent variable $\mathbf{c}_t$ induce distinct influences on the transition dynamics, specifically on the mapping from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$. To formalize this, we consider the operator k:

$$k(\mathbf{x}_t, \bar{\mathbf{x}}_t, \mathbf{x}_{t-1}, \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t) = \frac{p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t) \cdot p(\bar{\mathbf{x}}_t \mid \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t)}{p(\bar{\mathbf{x}}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t) \cdot p(\mathbf{x}_t \mid \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t)}, \tag{A1}$$

which separates into two multiplicative components:

$$k_1 = \frac{p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t)}{p(\mathbf{x}_t \mid \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t)}, \tag{A2}$$

$$k_2 = \frac{p(\bar{\mathbf{x}}_t \mid \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t)}{p(\bar{\mathbf{x}}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t)}. \tag{A3}$$

Here, k_1 and k_2 measure how changes in historical inputs affect the transition distribution at the current time step. The assumption requires that for any two distinct values of $\mathbf{c}_t$, the corresponding

operator k is different, indicating that the latent variable has a sufficiently strong influence on the system dynamics.

Since $\bar{\mathbf{x}}$ is in the neighborhood of $\mathbf{x}$, this formulation effectively captures second-order changes in the transition dynamics with respect to the latent variable $\mathbf{c}_t$. This reflects many real-world RL systems, where even unobservable latent factors (e.g., wind speed or goal target) cause noticeable and structured changes in transition behavior over time, for instance, by considering velocity as states.

In summary, these two assumptions are not only theoretically necessary for identification, but also naturally hold in many RL and control systems. They justify *the need to explicitly model and identify latent variables*, as such variables often induce meaningful and structured changes in both dynamics and optimal decision-making behavior.

Q5: On the Identification of Posterior Distribution and up to the Invertible Function (Theorem 1). *Why do we aim to identify the posterior distribution over latent variables, and what is the role of the invertible function h between the estimated and true latents?*

Theorem 1 establishes that the posterior distribution over latent factors given surrounding observations, $p(\mathbf{c}_t \mid \mathbf{x}_{t-2:t+1})$, is identifiable up to an invertible transformation. That is, the estimated latent $\hat{\mathbf{c}}_t$ satisfies $\hat{\mathbf{c}}_t = h(\mathbf{c}_t)$ for some invertible function h.

This form of identifiability is sufficient for downstream tasks such as dynamics modeling, planning, and control. Specifically, the learned dynamics or policy can be composed with h^{-1} without loss of expressiveness or utility. Since we only need to condition on the inferred latent $\hat{\mathbf{c}}_t$ to perform these tasks, any invertible transformation of the latent space preserves the representational capacity required for decision-making. In other words, although we may not recover the true latent variable $\mathbf{c}_t$ exactly, the recovered representation $\hat{\mathbf{c}}_t$ contains the same information and can be used equivalently in practice.

Therefore, identifying the posterior distribution (up to an invertible transformation) is both theoretically meaningful and practically sufficient for learning accurate dynamics models and optimal policies.

A.4 Discussions on the Model Design

Q6: On Different Settings (Planning and Policy). *How is `Ada-Diffuser` applied to both planning and policy learning settings?*

`Ada-Diffuser` is designed as a unified and generic framework that accommodates different types of inputs $\mathbf{x}$ (e.g., states, state-action pairs) and outputs (e.g., actions, trajectories, or state sequences). This flexibility allows it to support a wide range of planning and policy learning paradigms. We summarize four representative settings below:

- **Planning with state-action generation:** The model generates both states and actions, with latent variables influencing dynamics or rewards. This setting aligns with prior work such as Diffuser [5].
- **Planning with state-only generation:** The model generates future states, and an inverse dynamics model is used to recover the corresponding actions. This setup follows Decision Diffuser [70].
- **Planning from action-free demonstrations:** Only state sequences are available, and latent variables are assumed to capture high-level behaviors or skills. This setting extends latent diffusion planning [77].
- **Policy learning:** The model generates actions conditioned on the current or recent history of states. This includes multi-step action generation (as in Diffusion Policy [7]) and one-step action generation (as in Implicit Diffusion Q-Learning, IDQL [83]). In both cases, latent factors may affect the underlying dynamics or rewards.

These diverse settings demonstrate the universality of our framework and highlight that uncovering latent structure is a broadly applicable and critical problem in generative decision-making.

Q7: On the Latent Identification. *How is Stage 1 (Latent Identification) trained, and does it introduce additional computational overhead?*

In Stage 1, we train the latent identification module using an offline dataset, as commonly done in offline RL and imitation learning tasks. Specifically, we employ a lightweight variational autoencoder (VAE) to optimize the ELBO defined in Section 4.2. Empirically, this stage introduces minimal computational overhead (Appendix I.2). We further provide an ablation study in Appendix I.2 showing the impact of the number of training samples on the effectiveness of the latent identification module.

Q8: On the Temporal Block Design. *How does this reflect Theorem 1, and why do we not use exactly four steps in practice?*

Our approach reflects the theoretical result in Theorem 1 by identifying latent variables using small temporal blocks in both Stage 1 and Stage 2. In Stage 1, we segment trajectories into local blocks and optimize the ELBO to learn the posterior over latent variables. In Stage 2, we apply block-wise refinement to improve the posterior estimates using both past and one-step future observations, making a more accurate identification than using the prior alone.

While Theorem 1 shows that four consecutive time steps are sufficient for identifiability in principle, we do not strictly limit the block size to four in practice. Empirically, we find that using slightly larger blocks (typically between 6 and 20 steps) leads to more stable optimization and better performance. Our ablations in Appendix I.1 show that without access to future observations, identifiability degrades, aligning with the theory.

We treat the "four-step" condition not as a strict architectural constraint but as a theoretical justification (sufficient condition) for using small temporal blocks. The optimal number of steps in practice may vary depending on data properties, task complexity, and model capacity.

Q9: On the Refinement Step. *Why is the refinement step necessary, how does it work, and does it introduce additional computational overhead?*

The refinement step is motivated by the identification theory, which suggests that incorporating the current and future observations (other than only using historical ones) allows the model to infer a more informative posterior over latent variables than relying on the prior alone. This posterior refinement helps the model better capture latent dynamics by leveraging richer temporal context.

During training, the refinement step encourages the model to extract meaningful information from the posterior. Since Stage 1 optimizes the ELBO, the learned prior is already aligned with the posterior to some extent. This prevents the prior from collapsing into a trivial solution. The refinement step builds on this by using the pre-trained prior while further improving inference through contrastive learning between prior and posterior samples.

Importantly, this procedure does not introduce significant computational overhead. As shown in Appendix I.1, the refinement uses the same denoising network with different latent inputs ($\mathbf{c}$) and adds only a lightweight contrastive loss, making it efficient in practice.

A.5 Overview

In this appendix, we first present the theoretical analysis in Section B, including the proof of Theorem 1 and accompanying discussion, followed by the ELBO derivation for `Ada-Diffuser`. In Section C, we provide an in-depth analysis of different types of MDPs and their interconnections. Section H details the full `Ada-Diffuser` algorithm, model architectures, and its relation to Bayesian filtering. Section E expands on related work, covering diffusion-based decision-making, latent state estimation via belief learning, and autoregressive diffusion models. Finally, Sections F, G, H, and I provide additional details on benchmarks, baseline implementations, and complete experimental results.

B Theory

B.1 Notation List

We summarize the key notations used throughout the paper in Table A1, including variables for observed and latent states, temporal indices, and relevant mappings. These notations are used consistently in our theoretical analysis and algorithmic framework.

Index	Explanation	Support
$\mathbf{x}_t$	$[\mathbf{s}_t, \mathbf{a}_t]$, observed trajectories including state and action at time step t	$\mathcal{X}_t \subseteq \mathbb{R}^{d_a+d_s}$
d_x	dimension of observed variables	$d_a + d_s$
$\mathbf{s}_t$	state variable at time t	$\mathbf{s}_t \in \mathcal{S}_t$
$\mathbf{a}_t$	action variable at time t	$\mathbf{a}_t \in \mathcal{A}_t$
r_t	reward received at time t	$r_t \in \mathbb{R}$
$\mathbf{c}_t$	latent context variable at time t	$\mathbf{c}_t \in \mathcal{C}_t$
τ	trajectory sequence of $(\mathbf{s}_t, \mathbf{a}_t)$	$\{(\mathbf{s}_0, \mathbf{a}_0), \ldots, (\mathbf{s}_T, \mathbf{a}_T)\}$
$\boldsymbol{\tau}_x$	observable trajectory (states or state-actions)	$\boldsymbol{\tau}_{sa}$ or $\boldsymbol{\tau}_s$
$\boldsymbol{\tau}_c$	sequence of latent contexts	$\{\mathbf{c}_0, \ldots, \mathbf{c}_T\}$
$\boldsymbol{\tau}$	augmented trajectory with context	$[\boldsymbol{\tau}_x, \boldsymbol{\tau}_c]$
Function		
$\mathcal{T}$	transition dynamics conditioned on $\mathbf{c}_t$	$\mathcal{T}(\mathbf{s}_t \mid \mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{c}_t)$
$\mathcal{R}$	reward function conditioned on state, action, and context	$\mathcal{R}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{c}_t)$
π^E	expert policy used for generating demonstrations	$\pi^E(\mathbf{s}_t, \mathbf{c}_t)$
q_ψ	variational posterior for latent inference	$q_\psi(\mathbf{c}_t \mid \mathbf{x}_{t-T_x:t+1})$
p_ϕ	latent prior distribution	$p_\phi(\mathbf{c}_t \mid \mathbf{c}_{t-1})$
p_θ	generative model for transitions	$p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t)$
ϵ_θ	denoising network in diffusion process	$\epsilon_\theta(\cdot)$
Symbol		
$\eta_t, \epsilon_t, \delta_t$	exogenous noise in latent dynamics, state transitions, and reward	i.i.d. samples from noise distributions
$L_{a\|b}$	distribution operator from b to a	defined in Dunford and Schwartz [67]
$k(\cdot)$	ratio of joint probabilities used in uniqueness assumption	defined in Eq. 1
$\bar{\alpha}_t$	cumulative noise schedule in diffusion	product of forward noise factors
K	maximum number of diffusion steps	$K \in \mathbb{N}$
T_p, T_a	planning and action generation horizons	$T_p, T_a \in \mathbb{N}$
T_x	temporal block size for latent inference	$T_x \in \mathbb{N}$

Table A1: List of notations, explanations, and corresponding definitions.

Also, we formally define the operators used in the following.

Definition 1 (**Linear Operator** [67])**.** *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be random variables with supports* $\mathcal{A}$ *and* $\mathcal{B}$*, respectively. The linear operator* $L_{\mathbf{b}|\mathbf{a}}$ *is defined as a mapping from a probability function* $p_\mathbf{a} \in \mathcal{F}(\mathcal{A})$ *to a probability function* $p_\mathbf{b} \in \mathcal{F}(\mathcal{B})$*, given by*

$$\mathcal{F}(\mathcal{A}) \to \mathcal{F}(\mathcal{B}): \quad p_\mathbf{b} = L_{\mathbf{b}|\mathbf{a}} \circ p_\mathbf{a} = \int_\mathcal{A} p_{\mathbf{b}|\mathbf{a}}(\cdot \mid \mathbf{a})\, p_\mathbf{a}(\mathbf{a})\, d\mathbf{a}. \tag{A4}$$

Intuitively, this operator characterizes the transformation of probability distributions induced by the conditional distribution $p_{\mathbf{b}|\mathbf{a}}$. It provides a general representation of distributional change from $\mathbf{a}$ to $\mathbf{b}$, without imposing any parametric assumptions on the underlying distributions.

Definition 2 (**Diagonal Operator**)**.** *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be random variables with associated density functions* $p_\mathbf{a}$ *and* $p_\mathbf{b}$ *defined on supports* $\mathcal{A}$ *and* $\mathcal{B}$*, respectively. For a fixed value* $\mathbf{b} \in \mathcal{B}$*, the diagonal operator* $D_{\mathbf{b}|\mathbf{a}}$ *is defined as a linear operator that maps a density function* $p_\mathbf{a} \in \mathcal{F}(\mathcal{A})$ *to a function in* $\mathcal{F}(\mathcal{A})$ *via pointwise multiplication:*

$$D_{\mathbf{b}|\mathbf{a}} \circ p_\mathbf{a} = p_{\mathbf{b}|\mathbf{a}}(\mathbf{b} \mid \cdot) \cdot p_\mathbf{a}, \tag{A5}$$

where $D_{\mathbf{b}|\mathbf{a}} = p_{\mathbf{b}|\mathbf{a}}(\mathbf{b} \mid \cdot)$ *acts as a multiplication operator indexed by* $\mathbf{b}$*.*

B.2 Proof of Theorem 1

Proof. By the definition of data generation process (Fig. 1), the observed density is represented by:

$$
\begin{aligned}
& p_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} \\
&= \int_{\mathcal{C}_t}\int_{\mathcal{C}_{t-1}} p_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{c}_t,\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}}\, d\mathbf{c}_t d\mathbf{c}_{t-1} \\
&= \int_{\mathcal{C}_t}\int_{\mathcal{C}_{t-1}} p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2},\mathbf{c}_t,\mathbf{c}_{t-1}} p_{\mathbf{x}_t,\mathbf{c}_t|\mathbf{x}_{t-1},\mathbf{x}_{t-2},\mathbf{c}_{t-1}} p_{\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}}\, d\mathbf{c}_t d\mathbf{c}_{t-1} \\
&= \int_{\mathcal{C}_t}\int_{\mathcal{C}_{t-1}} p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} p_{\mathbf{x}_t,\mathbf{c}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}} p_{\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}}\, d\mathbf{c}_t d\mathbf{c}_{t-1} \\
&= \int_{\mathcal{C}_t}\int_{\mathcal{C}_{t-1}} p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t,\mathbf{c}_{t-1}} p_{\mathbf{c}_t|\mathbf{x}_{t-1},\mathbf{x}_{t-2},\mathbf{c}_{t-1}} p_{\mathbf{x}_{t-1},\mathbf{x}_{t-2},\mathbf{c}_{t-1}}\, d\mathbf{c}_t d\mathbf{c}_{t-1}. \\
&= \int_{\mathcal{C}_t}\int_{\mathcal{C}_{t-1}} p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t,\mathbf{c}_{t-1}} p_{\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2},\mathbf{c}_{t-1}}\, d\mathbf{c}_t d\mathbf{c}_{t-1}.
\end{aligned}
$$

Subsequently, the property of Markov process presents conditional independence, which is organized as follows:

$$
\begin{aligned}
p_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} &= \int_{\mathcal{C}_t} p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t} \left(\int_{\mathcal{C}_{t-1}} p_{\mathbf{c}_t,\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}}\, d\mathbf{c}_{t-1} \right) d\mathbf{c}_t \\
&= \int_{\mathcal{C}_t} p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t} p_{\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}\, d\mathbf{c}_t. \qquad \text{(A6)}
\end{aligned}
$$

Eq. A6 can be denoted in terms of operators: given values of $(\mathbf{x}_t, \mathbf{x}_{t-1}) \in \mathcal{X}_t \times \mathcal{X}_{t-1}$, this is

$$
L_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} = L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} D_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t} L_{\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}. \qquad \text{(A7)}
$$

Eq. A7 is the operator representation of the observed density function in 4 measurements.

Furthermore, the structure of Markov process implies the following two equalities:

$$
\begin{aligned}
p_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} &= \int_{\mathcal{C}_t} p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} p_{\mathbf{x}_t,\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}\, d\mathbf{c}_t, \\
p_{\mathbf{x}_t,\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} &= \int_{\mathcal{C}_{t-1}} p_{\mathbf{x}_t,\mathbf{c}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}} p_{\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}}\, d\mathbf{c}_{t-1}. \qquad \text{(A8)}
\end{aligned}
$$

For any fixed $(\mathbf{x}_t, \mathbf{x}_{t-1}) \in \mathcal{X}_t \times \mathcal{X}_{t-1}$, we notate Eq. A8 in terms of operators as follows:

$$
\begin{aligned}
L_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} &= L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} L_{\mathbf{x}_t,\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}, \\
L_{\mathbf{x}_t,\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} &= L_{\mathbf{x}_t,\mathbf{c}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}} L_{\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}}. \qquad \text{(A9)}
\end{aligned}
$$

Substituting the second line in Eq. A9 into R.H.S. of the first equation, we obtain

$$
\begin{aligned}
L_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} &= L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} L_{\mathbf{x}_t,\mathbf{c}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}} L_{\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}} \\
\Leftrightarrow L_{\mathbf{x}_t,\mathbf{c}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}} L_{\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}} &= L^{-1}_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} L_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}. \qquad \text{(A10)}
\end{aligned}
$$

The second line above uses Assumption 2 that $L^{-1}_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}$ is injective. Next, we show how to eliminate $L_{\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}}$ from the above. Consider 3 measurements $\{\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}\}$, we have

$$
p_{\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} = \int_{\mathcal{C}_{t-1}} p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}} p_{\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}}\, d\mathbf{c}_{t-1}, \qquad \text{(A11)}
$$

which, in operator notation (for fixed $\mathbf{x}_{t-1}$), is denoted as

$$
\begin{aligned}
L_{\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} &= L_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}} L_{\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}}, \\
\Rightarrow \qquad L_{\mathbf{c}_{t-1},\mathbf{x}_{t-1},\mathbf{x}_{t-2}} &= L^{-1}_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}} L_{\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}. \qquad \text{(A12)}
\end{aligned}
$$

The R.H.S. applies Assumption 2. Hence, substituting the above into Eq. A10, we obtain the desired representation:

$$L_{\mathbf{x}_t,\mathbf{c}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}} L^{-1}_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}} L_{\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} = L^{-1}_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} L_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}$$

$$\Rightarrow \quad L_{\mathbf{x}_t,\mathbf{c}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}} = L^{-1}_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} L_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} L^{-1}_{\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} L_{\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{c}_{t-1}}. \tag{A13}$$

The second line applies Assumption 2 to post-multiply by $L^{-1}_{\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}$, while in the third line, we postmultiply both sides by $L_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_{t-1}}$.

For each $\mathbf{x}_t$, choose a $\mathbf{x}_{t-1}$ and a neighborhood $\mathcal{N}^r$ around $(\mathbf{x}_t, \mathbf{x}_{t-1})$ to satisfy Assumption 2, and pick a $(\bar{\mathbf{x}}_t, \bar{\mathbf{x}}_{t-1})$ within the neighborhood $\mathcal{N}^r$ to satisfy Assumption 2. Because $(\bar{\mathbf{x}}_t, \bar{\mathbf{x}}_{t-1}) \in \mathcal{N}^r$, we also know that $(\mathbf{x}_t, \bar{\mathbf{x}}_{t-1})$, $(\bar{\mathbf{x}}_t, \mathbf{x}_{t-1}) \in \mathcal{N}^r$. The joint distribution of of observations can be represented by Eq. A7:

$$L_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} = L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} D_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t} L_{\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}. \tag{A14}$$

The first term on the R.H.S., $L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}$, does not depend on $\mathbf{x}_{t-1}$, and the last term $L_{\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}$ does not depend on $\mathbf{x}_t$. This feature suggests that, by evaluating Eq. A7 at the four pairs of points $(\mathbf{x}_t, \mathbf{x}_{t-1})$, $(\bar{\mathbf{x}}_t, \mathbf{x}_{t-1})$, $(\mathbf{x}_t, \bar{\mathbf{x}}_{t-1})$, $(\bar{\mathbf{x}}_t, \bar{\mathbf{x}}_{t-1})$, each pair of equations will share the same operator representation in common. Specifically:

$$L_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} = L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} D_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t} L_{\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}, \tag{A15}$$

$$L_{\mathbf{x}_{t+1},\bar{\mathbf{x}}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} = L_{\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t,\mathbf{c}_t} D_{\bar{\mathbf{x}}_t|\mathbf{x}_{t-1},\mathbf{c}_t} L_{\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}, \tag{A16}$$

$$L_{\mathbf{x}_{t+1},\mathbf{x}_t,\bar{\mathbf{x}}_{t-1},\mathbf{x}_{t-2}} = L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} D_{\mathbf{x}_t|\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} L_{\mathbf{c}_t,\bar{\mathbf{x}}_{t-1},\mathbf{x}_{t-2}}, \tag{A17}$$

$$L_{\mathbf{x}_{t+1},\bar{\mathbf{x}}_t,\bar{\mathbf{x}}_{t-1},\mathbf{x}_{t-2}} = L_{\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t,\mathbf{c}_t} D_{\bar{\mathbf{x}}_t|\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} L_{\mathbf{c}_t,\bar{\mathbf{x}}_{t-1},\mathbf{x}_{t-2}}. \tag{A18}$$

Assumption 2 implies that $L_{\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t,\mathbf{c}_t}$ is injective. Moreover, Assumption 3 implies $p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t) > 0$ for all $\mathbf{c}_t$, so that $D_{\bar{\mathbf{x}}_t|\mathbf{x}_{t-1},\mathbf{c}_t}$ is invertible. We can then solve for $L_{\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}$ from Eq. A16 as

$$D^{-1}_{\bar{\mathbf{x}}_t|\mathbf{x}_{t-1},\mathbf{c}_t} L^{-1}_{\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t,\mathbf{c}_t} L_{\mathbf{x}_{t+1},\bar{\mathbf{x}}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} = L_{\mathbf{c}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}. \tag{A19}$$

Plugging this expression into Eq. A15 leads to

$$L_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} = L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} D_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t} D^{-1}_{\bar{\mathbf{x}}_t|\mathbf{x}_{t-1},\mathbf{c}_t} L^{-1}_{\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t,\mathbf{c}_t} L_{\mathbf{x}_{t+1},\bar{\mathbf{x}}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}. \tag{A20}$$

At this point, we have decomposed the observable joint operator and expressed it in terms of latent-conditioned transitions, enabling spectral analysis for identifying latent structure.

Lemma 1 of [68] shows that, given the injectivity of $L_{\mathbf{x}_{t-2},\bar{\mathbf{x}}_{t-1},\mathbf{x}_t,\mathbf{x}_{t+1}}$ as in Assumption 2, we can postmultiply by $L^{-1}_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}}$ to obtain:

$$\mathbf{M} \equiv L_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} L^{-1}_{\mathbf{x}_{t+1},\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} = L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} D_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t} D^{-1}_{\bar{\mathbf{x}}_t|\mathbf{x}_{t-1},\mathbf{c}_t} L^{-1}_{\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t,\mathbf{c}_t}. \tag{A21}$$

Similarly, manipulations of Eq. A17 and A18 lead to

$$\mathbf{N} \equiv L_{\mathbf{x}_{t+1},\bar{\mathbf{x}}_t,\mathbf{x}_{t-1},\mathbf{x}_{t-2}} L^{-1}_{\mathbf{x}_{t+1},\mathbf{x}_t,\bar{\mathbf{x}}_{t-1},\mathbf{x}_{t-2}} = L_{\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t,\mathbf{c}_t} D_{\bar{\mathbf{x}}_t|\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} D^{-1}_{\mathbf{x}_t|\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} L^{-1}_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}. \tag{A22}$$

Assumption 2 guarantees that, for any $\mathbf{x}_t$, $(\bar{\mathbf{x}}_t, \mathbf{x}_{t-1}, \bar{\mathbf{x}}_{t-1})$ exist so that Eq. A21 and Eq. A22 are valid operations. Finally, we postmultiply Eq. A21 by Eq. A22 to obtain:

$$\begin{aligned}
\mathbf{MN} &= L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} D_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t} D^{-1}_{\bar{\mathbf{x}}_t|\mathbf{x}_{t-1},\mathbf{c}_t} \left(L_{\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t,\mathbf{c}_t} L_{\mathbf{x}_{t+1}|\bar{\mathbf{x}}_t,\mathbf{c}_t}\right) \times D_{\bar{\mathbf{x}}_t|\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} D^{-1}_{\mathbf{x}_t|\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} L^{-1}_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} \\
&= L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} \left(D_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t} D^{-1}_{\bar{\mathbf{x}}_t|\mathbf{x}_{t-1},\mathbf{c}_t} D_{\bar{\mathbf{x}}_t|\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} D^{-1}_{\mathbf{x}_t|\bar{\mathbf{x}}_{t-1},\mathbf{c}_t}\right) L^{-1}_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} \\
&\equiv L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} D_{\mathbf{x}_t,\bar{\mathbf{x}}_t,\mathbf{x}_{t-1},\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} L^{-1}_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t},
\end{aligned} \tag{A23}$$

where

$$\begin{aligned}
\left(D_{\mathbf{x}_t,\bar{\mathbf{x}}_t,\mathbf{x}_{t-1},\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} h\right)(\mathbf{c}_t) &= \left(D_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t} D^{-1}_{\bar{\mathbf{x}}_t|\mathbf{x}_{t-1},\mathbf{c}_t} D_{\bar{\mathbf{x}}_t|\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} D^{-1}_{\mathbf{x}_t|\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} h\right)(\mathbf{c}_t) \\
&= \frac{p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t) p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t}(\bar{\mathbf{x}}_t \mid \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t)}{p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t}(\bar{\mathbf{x}}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t) p_{\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{c}_t}(\mathbf{x}_t \mid \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t)} h(\mathbf{c}_t) \\
&\equiv k(\mathbf{x}_t, \bar{\mathbf{x}}_t, \mathbf{x}_{t-1}, \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t) h(\mathbf{c}_t).
\end{aligned} \tag{A24}$$

This equation implies that the observed operator $\mathbf{MN}$ on the L.H.S. of Eq. A25 has an inherent eigenvalue–eigenfunction decomposition, with the eigenvalues corresponding to the function $k(\mathbf{x}_t, \bar{\mathbf{x}}_t, \mathbf{x}_{t-1}, \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t)$ and the eigenfunctions corresponding to the density $p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid \mathbf{x}_t, \mathbf{c}_t)$.

The decomposition in Eq. A25 is similar to the decomposition in nonparametric identification [68, 84]. First, Assumption 3 ensures this decomposition is unique. Second, the operator $\mathbf{MN}$ on the L.H.S. has the same spectrum as the diagonal operator $D_{\mathbf{x}_t,\bar{\mathbf{x}}_t,\mathbf{x}_{t-1},\bar{\mathbf{x}}_{t-1},\mathbf{c}_t}$. Assumption 3 guarantees that the spectrum of the diagonal operator is bounded. Since an operator is bounded by the largest element of its spectrum, Assumption 3 also implies that the operator $\mathbf{MN}$ is bounded, whence we can apply Theorem XV.4.3.5 from [67] to show the uniqueness of the spectral decomposition of bounded linear operators:

$$L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} = C L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t} P^{-1}. \quad D_{\mathbf{x}_t,\bar{\mathbf{x}}_t,\mathbf{x}_{t-1},\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} = P D_{\mathbf{x}_t,\bar{\mathbf{x}}_t,\mathbf{x}_{t-1},\bar{\mathbf{x}}_{t-1},\mathbf{c}_t} P^{-1} \tag{A25}$$

where C is a scalar accounting for scaling indeterminacy and P is a permutation on the order of elements in $D_{\hat{\mathbf{x}}_t|\hat{\mathbf{c}}_t}$, as discussed in [67]. These forms of indeterminacy are analogous to those in eigendecomposition, which can be viewed as a finite-dimensional special case.

We will show why the uniqueness of spectral decomposition is informative. First, since the normalizing condition,

$$\int_{\hat{\mathcal{X}}_{t+1}} p_{\hat{\mathbf{x}}_{t+1}|\hat{\mathbf{x}}_t,\hat{\mathbf{c}}_t} \, d\hat{\mathbf{x}}_{t+1} = 1 \tag{A26}$$

must hold for every $\hat{\mathbf{c}}_t$, one only solution is to set $C = 1$.

Second, Assumption 3 implies that Eq. A25 imply that the eigenvalues $k(\mathbf{x}_t, \bar{\mathbf{x}}_t, \mathbf{x}_{t-1}, \bar{\mathbf{x}}_{t-1}, \mathbf{c}_t)$ are distinct for different values $\mathbf{c}_t$. When the eigenvalues are the same for multiple values of $\mathbf{c}_t$, the corresponding eigenfunctions are only determined up to an arbitrary linear combination, implying that they are not identified. It also implies that for each $\mathbf{x}_t$, we can find values $\bar{\mathbf{x}}_t, \mathbf{x}_{t-1}, \bar{\mathbf{x}}_{t-1}$ such that the eigenvalues are distinct across all $\mathbf{c}_t$.

Ultimately, the unorder of eigenvalues/eigenfunctions is left. The operator, $L_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}$, corresponding to the set $\{p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid \mathbf{x}_t, \mathbf{c}_t)\}$ for all $\mathbf{x}_t, \mathbf{c}_t$, admits a unique solution (orderibng ambiguity of eigendecomposition only changes the entry position):

$$\{p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid \mathbf{x}_t, \mathbf{c}_t)\} = \{p_{\mathbf{x}_{t+1}|\hat{\mathbf{x}}_t,\hat{\mathbf{c}}_t}(\mathbf{x}_{t+1} \mid \hat{\mathbf{x}}_t, \hat{\mathbf{c}}_t)\}, \qquad \text{for all } \mathbf{x}_t, \mathbf{c}_t, \hat{\mathbf{x}}_t, \hat{\mathbf{c}}_t \tag{A27}$$

Due to the set is unorder, the only way to match the R.H.S. with the L.H.S. in a consistent order is to exchange the conditioning variables, that is,

$$\begin{aligned} &\{p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid \mathbf{x}_t^{(1)}, \mathbf{c}_t^{(1)}),\, p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid \mathbf{x}_t^{(2)}, \mathbf{c}_t^{(2)}),\, \ldots\} \\ &= \{p_{\mathbf{x}_{t+1}|\hat{\mathbf{x}}_t,\hat{\mathbf{c}}_t}(\cdot \mid \hat{\mathbf{x}}_t^{(1)}, \hat{\mathbf{c}}_t^{(1)}),\, p_{\mathbf{x}_{t+1}|\hat{\mathbf{x}}_t,\hat{\mathbf{c}}_t}(\cdot \mid \hat{\mathbf{x}}_t^{(2)}, \hat{\mathbf{c}}_t^{(2)}),\, \ldots\} \\ \Rightarrow \quad &[p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid \mathbf{x}_t^{(\pi(1))}, \mathbf{c}_t^{(\pi(1))}),\, p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid \mathbf{x}_t^{(\pi(2))}, \mathbf{c}_t^{(\pi(2))}),\, \ldots] \\ &= [p_{\mathbf{x}_{t+1}|\hat{\mathbf{x}}_t,\hat{\mathbf{c}}_t}(\cdot \mid \hat{\mathbf{x}}_t^{(\pi(1))}, \hat{\mathbf{c}}_t^{(\pi(1))}),\, p_{\mathbf{x}_{t+1}|\hat{\mathbf{x}}_t,\hat{\mathbf{c}}_t}(\cdot \mid \hat{\mathbf{x}}_t^{(\pi(2))}, \hat{\mathbf{c}}_t^{(\pi(2))}),\, \ldots] \end{aligned} \tag{A28}$$

where superscript $(\cdot)$ denotes the index of the conditioning variables $[\mathbf{x}_t, \mathbf{c}_t]$, and π is reindexing the conditioning variables. We use a relabeling map H to represent its corresponding value mapping:

$$p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid H(\mathbf{x}_t, \mathbf{c}_t)) = p_{\mathbf{x}_{t+1}|\hat{\mathbf{x}}_t,\hat{\mathbf{c}}_t}(\cdot \mid \hat{\mathbf{x}}_t, \hat{\mathbf{c}}_t), \qquad \text{for all } \mathbf{x}_t, \mathbf{c}_t, \hat{\mathbf{x}}_t, \hat{\mathbf{c}}_t \tag{A29}$$

By Assumption 3, different x^* corresponds to different $p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid H(\mathbf{x}_t, \mathbf{c}_t))$, there is no repeated element in $\{p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid H(\mathbf{x}_t, \mathbf{c}_t))\}$ (and $\{p_{\mathbf{x}_{t+1}|\hat{\mathbf{x}}_t,\hat{\mathbf{c}}_t}(\cdot \mid \hat{\mathbf{x}}_t, \hat{\mathbf{c}}_t)\}$). Hence, the relabelling map H is one-to-one.

Furthermore, Assumption 3 implies that $p_{\mathbf{x}_{t+1},|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid H(\mathbf{x}_t, \mathbf{c}_t))$ determines a unique $H(\mathbf{x}_t, \mathbf{c}_t)$. The same holds for the $p_{\mathbf{x}_{t+1}|\hat{\mathbf{x}}_t,\hat{\mathbf{c}}_t}(\cdot \mid \hat{\mathbf{x}}_t, \hat{\mathbf{c}}_t)$, implying that

$$p_{\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_t}(\cdot \mid H(\mathbf{x}_t, \mathbf{c}_t)) = p_{\mathbf{x}_{t+1}|\hat{\mathbf{x}}_t,\hat{\mathbf{c}}_t}(\cdot \mid \hat{\mathbf{x}}_t, \hat{\mathbf{c}}_t) \implies \hat{\mathbf{x}}_t, \hat{\mathbf{c}}_t = H(\mathbf{x}_t, \mathbf{c}_t) \tag{A30}$$

Since the observation $\mathbf{x}_t$ is known and suppose $\hat{\mathbf{x}}_t = \mathbf{x}_t$, this relationship indeed represents an invertible transformation between $\hat{\mathbf{c}}_t$ and $\mathbf{c}_t$ as

$$\hat{\mathbf{c}}_t = h(\mathbf{c}_t). \tag{A31}$$

which ensures that $p(\mathbf{c}_t \mid \mathbf{x}_{t-2:t+1})$ can be identifiable up to an invertible transformation on the latent variables $\hat{\mathbf{c}}_t = h(\mathbf{c}_t)$ □

B.3 Discussion on Assumptions

Assumption 2. The assumption of the injectivity of a linear operator is commonly employed in the nonparametric identification [68, 84, 69]. Intuitively, it means that different input distributions of a linear operator correspond to different output distributions of that operator. For a better understanding of this assumption, we provide several examples that describe the mapping from $p_a \Rightarrow p_b$, where a and b are random variables:

Example 1 (Inverse Nonlinear Transformation)**.** $b = g(a)$*, where* g *is an invertible function.*

Example 2 (Additive Transformation)**.** $b = a + \epsilon$*, where* $p(\epsilon)$ *must not vanish everywhere after the Fourier transform.*

Example 3 (Nonlinear Additive Transformation)**.** $b = g(a) + \epsilon$*, where the same conditions from Examples 1 and 2 are required.*

Example 4 (Post-linear Transformation)**.** $b = g_1(g_2(a) + \epsilon)$*, a post-nonlinear model with invertible nonlinear functions* g_1, g_2*, combining the assumptions in* ***Examples 1-3****.*

Example 5 (Nonlinear Transformation with Exponential Family)**.** $b = g(a, \epsilon)$*, where the joint distribution* $p(a, b)$ *follows an exponential family.*

Example 6 (General Nonlinear Transformation)**.** $b = g(a, \epsilon)$*, a general nonlinear formulation. Certain deviations from the nonlinear additive model (****Example*** *3), e.g., polynomial perturbations, can still be tractable.*

B.4 ELBO of `Ada-Diffuser`

In this section, we provide analysis on the $\mathbf{x}^0$-prediction Mean Squared Error (MSE) loss objectives used in the Denoise-and-Refine Mechanism of `Ada-Diffuser`. Our main argument establishes that minimizing the reconstruction losses $\mathcal{L}_{\text{prior}}$ and $\mathcal{L}_{\text{post}}$ corresponds to optimizing an ELBO on the conditional log-likelihood of the clean observation $\mathbf{x}_t^0$, given a noisy observation $\mathbf{x}_t^k$ and an inferred latent context $\mathbf{c}_t$.

Let $\mathbf{x}_t^0 \sim q(\mathbf{x}_t^0)$ be a clean data sample from the true data distribution at sequence time step t. Let $\mathbf{c}_t$ be the inferred latent context relevant to $\mathbf{x}_t^0$.

The forward diffusion process gradually adds Gaussian noise to $\mathbf{x}_t^0$ over K diffusion steps:

$$q(\mathbf{x}_t^k|\mathbf{x}_t^{k-1}) = \mathcal{N}(\mathbf{x}_t^k; \sqrt{\alpha_k}\mathbf{x}_t^{k-1}, (1-\alpha_k)\mathbf{I})$$

for $k \in \{1, ..., K\}$, where $\alpha_k \in (0, 1)$ are predefined noise schedule parameters. This process allows sampling $\mathbf{x}_t^k$ directly from $\mathbf{x}_t^0$:

$$\mathbf{x}_t^k = \sqrt{\bar{\alpha}_k}\mathbf{x}_t^0 + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \text{ and } \bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i.$$

The reverse process $p_\theta(\mathbf{x}_t^{k-1}|\mathbf{x}_t^k, \mathbf{c}_t)$ that parameterized by θ aims to denoise $\mathbf{x}_t^k$ to $\mathbf{x}_t^{k-1}$ conditioned on $\mathbf{c}_t$.

The derivation of the ELBO for diffusion models is standard following DDPM related derivations [60, 48]. The conditional log-likelihood $\log p_\theta(\mathbf{x}_t^0|\mathbf{c}_t)$ can be lower-bounded using the ELBO:

$$\log p_\theta(\mathbf{x}_t^0|\mathbf{c}_t) \geq \mathbb{E}_{q(\mathbf{x}_t^{1:K}|\mathbf{x}_t^0)}\left[\log p_\theta(\mathbf{x}_t^K|\mathbf{c}_t) + \sum_{k=1}^{K} \log \frac{p_\theta(\mathbf{x}_t^{k-1}|\mathbf{x}_t^k, \mathbf{c}_t)}{q(\mathbf{x}_t^{k-1}|\mathbf{x}_t^k, \mathbf{x}_t^0)}\right]$$

Assuming p_θ satisfies Markov Property (i.e., $p_\theta(\mathbf{x}_t^{k-1}|\mathbf{x}_t^k, \ldots, \mathbf{x}_t^K, \mathbf{c}_t) = p_\theta(\mathbf{x}_t^{k-1}|\mathbf{x}_t^k, \mathbf{c}_t)$), which is a standard structural assumption for diffusion models, the ELBO can be rewritten as:

$$\begin{aligned}
\log p_\theta(\mathbf{x}_t^0|\mathbf{c}_t) \geq & \underbrace{\mathbb{E}_{q(\mathbf{x}_t^1|\mathbf{x}_t^0)}[\log p_\theta(\mathbf{x}_t^0|\mathbf{x}_t^1, \mathbf{c}_t)]}_{L_0} \\
& - \sum_{k=2}^{K} \underbrace{\mathbb{E}_{q(\mathbf{x}_t^k|\mathbf{x}_t^0)}[D_{KL}(q(\mathbf{x}_t^{k-1}|\mathbf{x}_t^k, \mathbf{x}_t^0)||p_\theta(\mathbf{x}_t^{k-1}|\mathbf{x}_t^k, \mathbf{c}_t))]}_{L_{k-1}} \\
& - \underbrace{D_{KL}(q(\mathbf{x}_t^K|\mathbf{x}_t^0)||p_\theta(\mathbf{x}_t^K|\mathbf{c}_t))}_{L_K},
\end{aligned}$$

This inequality holds with equality if and only if the model's true posterior over the latent diffusion path, $p_\theta(\mathbf{x}_t^{1:K}|\mathbf{x}_t^0, \mathbf{c}_t)$, is identical to the approximate posterior used to derive the ELBO, which is the forward noising process $q(\mathbf{x}_t^{1:K}|\mathbf{x}_t^0)$. This bound can also include an additive constant $C(\mathbf{x}_t^0, \mathbf{c}_t)$ which does not depend on the model parameters θ and is thus typically omitted when focusing on terms relevant to parameter optimization.

To maximize $\log p_\theta(\mathbf{x}_t^0|\mathbf{c}_t)$, we aim to maximize this lower bound by optimizing L_0 (i.e., maximizing this term) and each L_{k-1} term (i.e., minimizing these D_{KL} terms, as they appear with a negative sign). The term L_K is often treated as a constant (or absorbed into $C(\mathbf{x}_t^0, \mathbf{c}_t)$) if $p_\theta(\mathbf{x}_t^K|\mathbf{c}_t)$ is set to a standard Gaussian $\mathcal{N}(0, \mathbf{I})$ and $\bar{\alpha}_K \approx 0$.

We parameterize the reverse process $p_\theta(\mathbf{x}_t^{k-1}|\mathbf{x}_t^k, \mathbf{c}_t)$ as a Gaussian:

$$p_\theta(\mathbf{x}_t^{k-1}|\mathbf{x}_t^k, \mathbf{c}_t) = \mathcal{N}(\mathbf{x}_t^{k-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t^k, k, \mathbf{c}_t), \sigma_k^2\mathbf{I})$$

The true posterior step $q(\mathbf{x}_t^{k-1}|\mathbf{x}_t^k, \mathbf{x}_t^0)$ is also Gaussian:

$$q(\mathbf{x}_t^{k-1}|\mathbf{x}_t^k, \mathbf{x}_t^0) = \mathcal{N}(\mathbf{x}_t^{k-1}; \tilde{\boldsymbol{\mu}}_k(\mathbf{x}_t^k, \mathbf{x}_t^0), \tilde{\sigma}_k^2\mathbf{I})$$

where $\tilde{\boldsymbol{\mu}}_k(\mathbf{x}_t^k, \mathbf{x}_t^0) = \frac{\sqrt{\bar{\alpha}_{k-1}}(1-\alpha_k)}{1-\bar{\alpha}_k}\mathbf{x}_t^0 + \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}\mathbf{x}_t^k$ and $\tilde{\sigma}_k^2 = \frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k}(1-\alpha_k)$ is the variance.

For an $\mathbf{x}^0$-prediction model, denoted as $\epsilon_\theta(\mathbf{x}_t^k, k, \mathbf{c}_t)$ in the main paper, that aims to predict $\mathbf{x}_t^0$ from the noisy input $\mathbf{x}_t^k$ and context $\mathbf{c}_t$, the mean of the reverse model $\boldsymbol{\mu}_\theta$ can be expressed as:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t^k, k, \mathbf{c}_t) = \frac{\sqrt{\bar{\alpha}_{k-1}}(1-\alpha_k)}{1-\bar{\alpha}_k}\epsilon_\theta(\mathbf{x}_t^k, k, \mathbf{c}_t) + \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}\mathbf{x}_t^k$$

Choosing $\sigma_k^2 = \tilde{\sigma}_k^2$, the KL divergence term L_{k-1} simplifies to:

$$\begin{aligned} L_{k-1} &= \mathbb{E}_{q(\mathbf{x}_t^k|\mathbf{x}_t^0)}\left[\frac{1}{2\sigma_k^2}\left\|\tilde{\boldsymbol{\mu}}_k(\mathbf{x}_t^k, \mathbf{x}_t^0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t^k, k, \mathbf{c}_t)\right\|^2\right] + C_k' \\ &= \mathbb{E}_{\mathbf{x}_t^0, \boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_k^2}\left(\frac{\sqrt{\bar{\alpha}_{k-1}}(1-\alpha_k)}{1-\bar{\alpha}_k}\right)^2\left\|\mathbf{x}_t^0 - \epsilon_\theta(\sqrt{\bar{\alpha}_k}\mathbf{x}_t^0 + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}, k, \mathbf{c}_t)\right\|^2\right] + C_k' \end{aligned}$$

where C_k' are constants not depending on θ. The expectation $\mathbb{E}_{\mathbf{x}_t^0, \boldsymbol{\epsilon}}$ denotes averaging over clean data $\mathbf{x}_t^0$ and the noise $\boldsymbol{\epsilon}$ used to construct $\mathbf{x}_t^k$. Thus, maximizing the ELBO contribution from $-L_{k-1}$ is equivalent to minimizing the following weighted MSE term:

$$\mathbb{E}_{\mathbf{x}_t^0, \boldsymbol{\epsilon}, \mathbf{c}_t}\left[w(k)\left\|\mathbf{x}_t^0 - \epsilon_\theta(\sqrt{\bar{\alpha}_k}\mathbf{x}_t^0 + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}, k, \mathbf{c}_t)\right\|^2\right] \tag{A32}$$

where $w(k) = \frac{1}{2\sigma_k^2}\left(\frac{\sqrt{\bar{\alpha}_{k-1}}(1-\alpha_k)}{1-\bar{\alpha}_k}\right)^2$ is a positive weighting factor.

The term $L_0 = \mathbb{E}_{q(\mathbf{x}_t^1|\mathbf{x}_t^0)}[\log p_\theta(\mathbf{x}_t^0|\mathbf{x}_t^1, \mathbf{c}_t)]$ can also be made proportional to an MSE if $p_\theta(\mathbf{x}_t^0|\mathbf{x}_t^1, \mathbf{c}_t)$ is a Gaussian centered at $\epsilon_\theta(\mathbf{x}_t^1, 1, \mathbf{c}_t)$:

$$\log p_\theta(\mathbf{x}_t^0|\mathbf{x}_t^1, \mathbf{c}_t) = -\frac{1}{2\sigma_1^2}\left\|\mathbf{x}_t^0 - \epsilon_\theta(\mathbf{x}_t^1, 1, \mathbf{c}_t)\right\|^2 + \text{const}$$

Maximizing L_0 is then equivalent to minimizing this MSE.

The diffusion model ϵ_θ is typically trained by minimizing a simplified objective (e.g., [60]), often an unweighted or equally weighted sum of these MSE terms over uniformly sampled diffusion steps $k \in [1, K]$ and data $\mathbf{x}_t^0$:

$$\mathcal{L}_{\text{simple}}(\theta) = \mathbb{E}_{k\sim U[1,K], \mathbf{x}_t^0, \boldsymbol{\epsilon}, \mathbf{c}_t}\left[\left\|\mathbf{x}_t^0 - \epsilon_\theta(\sqrt{\bar{\alpha}_k}\mathbf{x}_t^0 + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}, k, \mathbf{c}_t)\right\|^2\right]$$

This simplification is justified by arguing that reweighting terms $w(k)$ in Equation A32 can be absorbed into the network or do not significantly alter the optimal solution for expressive models, allowing $w(k)$ to be effectively set to 1.

The Denoise-and-Refine losses are:

$$\mathcal{L}_{\text{prior}} = \mathbb{E}_{\mathbf{x}_t^0, \boldsymbol{\epsilon}, \hat{\mathbf{c}}_t^{\text{prior}}}\left[\left\|\mathbf{x}_t^0 - \epsilon_\theta(\mathbf{x}_t^{k_i}, k_i, \hat{\mathbf{c}}_t^{\text{prior}})\right\|^2\right]$$

$$\mathcal{L}_{\text{post}} = \mathbb{E}_{\mathbf{x}_t^0, \boldsymbol{\epsilon}, \hat{\mathbf{c}}_t^{\text{post}}} \left[\left\| \mathbf{x}_t^0 - \epsilon_\theta(\mathbf{x}_t^{k_i}, k_i, \hat{\mathbf{c}}_t^{\text{post}}) \right\|^2 \right]$$

where $\mathbf{x}_t^{k_i} = \sqrt{\bar{\alpha}_{k_i}}\mathbf{x}_t^0 + \sqrt{1-\bar{\alpha}_{k_i}}\boldsymbol{\epsilon}$, and k_i is the specific input noise level for the observation $\mathbf{x}_t$ determined by the causal denoising schedule $k_i = \frac{i}{T}K$. These losses, $\mathcal{L}_{\text{prior}}$ and $\mathcal{L}_{\text{post}}$, are specific instances of the simplified MSE loss objective in equation A32 with $w(k_i) \approx 1$, conditioned on the inferred contexts $\hat{\mathbf{c}}_t^{\text{prior}}$ and $\hat{\mathbf{c}}_t^{\text{post}}$ respectively. Consequently, minimizing these MSE losses directly optimizes the corresponding terms in the ELBO for $\log p_\theta(\mathbf{x}_t^0|\mathbf{c}_t)$.

Therefore, we have proven that minimizing $\mathcal{L}_{\text{prior}}$ and $\mathcal{L}_{\text{post}}$ as defined in the Denoise-and-Refine mechanism serves to maximize a variational lower bound on the conditional log-likelihood $\log p_\theta(\mathbf{x}_t^0|\mathbf{c}_t)$. The underlying diffusion model $\epsilon_\theta(\cdot, k, \cdot)$ is trained to be proficient at denoising from a range of noise levels k, as captured by objectives such as $\mathcal{L}_{\text{simple}}$. The specific monotonically increasing noise schedule k_i used in $\mathcal{L}_{\text{prior}}$ and $\mathcal{L}_{\text{post}}$ represents a particular instance from this range of noise levels. Thus, these objectives are theoretically grounded in the principles of variational inference for diffusion models, adapted to conditioning on the inferred latent context $\mathbf{c}_t$ and applied at specific noise levels relevant to the autoregressive denoising process of `Ada-Diffuser`.

C Summary on Different MDPs

Our work considers a contextual POMDP setting with an evolving latent process, which naturally relates to several established MDP formulations, including contextual MDPs [85], hidden-parameter MDPs (HiP-MDPs) [62], and their variants. In this section, we provide formal definitions of these models and discuss their relationships and distinctions.

C.1 Contexutal MDPs

A contextual Markov decision process (CMDP) [85] is defined by the tuple $\langle \mathcal{C}, \mathcal{S}, \mathcal{A}, \mathcal{M} \rangle$, where $\mathcal{C}$ is the context space, $\mathcal{S}$ is the state space, and $\mathcal{A}$ is the action space. The mapping $\mathcal{M}$ assigns to each context $c \in \mathcal{C}$ a set of MDP parameters $\mathcal{M}(c) = \{R^c, T^c\}$, where R^c and T^c are the reward and transition functions associated with context c.

Sodhani et al. [86] and Liang et al. [45] extend the CMDP framework to settings in which the context variable $\mathbf{c}$ evolves according to its own Markovian dynamics $p(\mathbf{c}_{t+1} \mid \mathbf{c}_t)$, closely aligning with our formulation of a latent process evolving over time.

C.2 Hidden-Parameter MDPs

Hidden-Parameter MDPs (HiP-MDPs) [62] are defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \Theta, \mathcal{T}, \mathcal{R}, \gamma, P_\Theta \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and Θ is the space of task-specific latent parameters. For each $\theta \in \Theta$, the transition and reward functions are given by $\mathcal{T}_\theta : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ and $\mathcal{R}_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, respectively. The parameter θ is sampled from a prior distribution P_Θ at the beginning of an episode and remains fixed during the episode. The discount factor is denoted by $\gamma \in [0, 1)$. This framework defines a family of MDPs indexed by the latent parameter θ, with each θ inducing a different set of dynamics and reward functions. It can be seen as a special case of a contextual MDP where the context is latent and fixed per episode.Xie et al. [29] further generalize this framework by allowing the task parameter θ to evolve dynamically across episodes, rather than being fixed.

Bayes-Adaptive MDPs (BAMDPs) are closely related to both HiP-MDPs and contextual MDPs (CMDPs). In BAMDPs, the agent maintains a posterior distribution over MDPs based on its interaction history. Specifically, it maintains a belief $b_t(R, T) = p(R, T \mid \boldsymbol{\tau}_{:t})$, where $\tau_{:t} = \{\mathbf{s}_0, \mathbf{a}_0, r_0, \ldots, \mathbf{s}_t\}$ denotes the trajectory observed up to time t. This belief captures the agent's uncertainty about the underlying transition and reward functions.

The transition and reward functions can then be defined in expectation over this posterior, effectively conditioning decision-making on the belief b_t. When the environment is driven by hidden contextual variables or latent task parameters, such as in CMDPs or HiP-MDPs—this belief can be interpreted as a distribution over these latent variables. In this view, BAMDPs provide a non-parametric framework for reasoning over hidden structure, while approaches like ours explicitly model such latent variables

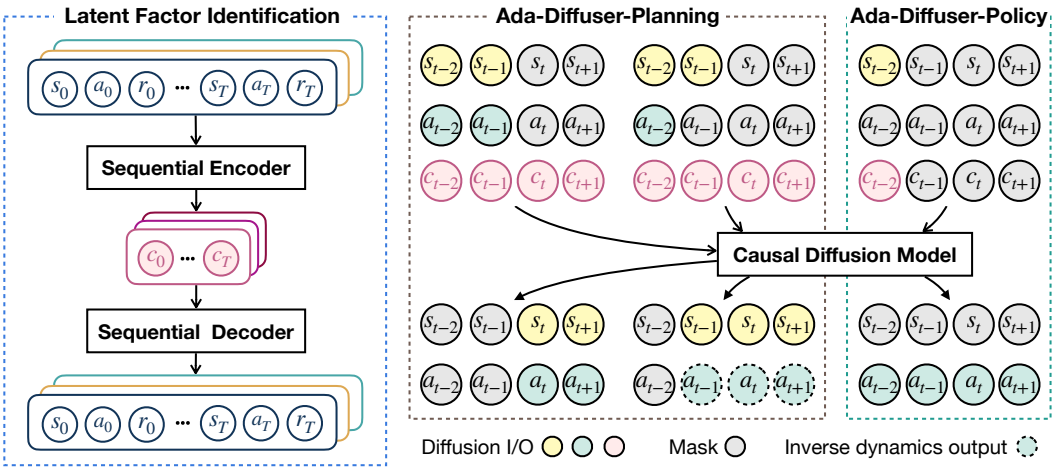

Figure A1: **Overview of the `Ada-Diffuser` framework.** The modular design consists of two main stages: latent context identification (Stage 1, Section 4.2), followed by a causal diffusion process (Stage 2, Section 4.3) that models the generative structure of the trajectories. The learned model is then used for planning or policy learning conditioned on the inferred latent context.

and infer their posterior distributions using amortized inference. Both aim to enable adaptive planning and learning under uncertainty, but differ in how latent structure is represented and inferred.

C.3 Discussions and Comparisons

The key distinction between contextual MDPs and hidden-parameter MDPs lies in how the latent factors are represented: contextual MDPs explicitly treat them as latent variables, while HiP-MDPs model them implicitly as parameters governing the transition and reward functions. In our work, we adopt the contextual MDP perspective, where the latent process is modeled as a random variable that evolves over time.

However, our identification theory, focused on recovering the posterior distribution over latent variables, also applies to the HiP-MDP setting. Once the posterior over the hidden parameters is identified, the corresponding transition and reward functions can be recovered as well.

Additionally, our framework, which models a factorization over observed states and latent variables, is conceptually related to factored MDPs [66]. In a factored MDP, the state space $\mathcal{S}$ is represented as a set of variables $\mathcal{S} = \{s^{(1)}, s^{(2)}, \ldots, s^{(n)}\}$, and the transition and reward functions are decomposed over these factors:

$$T(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) = \prod_{i=1}^{n} T_i \left(s'^{(i)} \mid \mathrm{Pa}_T^{(i)}(\mathbf{s}, \mathbf{a}) \right), \quad R(\mathbf{s}, \mathbf{a}) = \sum_{j=1}^{m} R_j \left(\mathrm{Pa}_R^{(j)}(\mathbf{s}, \mathbf{a}) \right),$$

where $\mathrm{Pa}_T^{(i)}$ and $\mathrm{Pa}_R^{(j)}$ denote the parent variables (i.e., dependencies) for each transition and reward component, respectively. our framework, while not relying on an explicit graphical structure, shares conceptual similarities with factored MDPs [66] through its coarse-grained factorization over observed states and latent variables. Specifically, we distinguish between latent variables that affect the transition dynamics and those that affect the reward function. Formally, we express the generative process as:

$$T(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t, \mathbf{c}_t^{\mathrm{s}}), \quad R(r_t \mid \mathbf{s}_t, \mathbf{a}_t, \mathbf{c}_t^{\mathrm{r}}),$$

where $\mathbf{c}_t^{\mathrm{s}}$ and $\mathbf{c}_t^{\mathrm{r}}$ are distinct (or potentially overlapping) latent factors that influence transitions and rewards, respectively. This separation enables flexible modeling of partially observable environments where different unobserved processes govern the dynamics and task objectives.

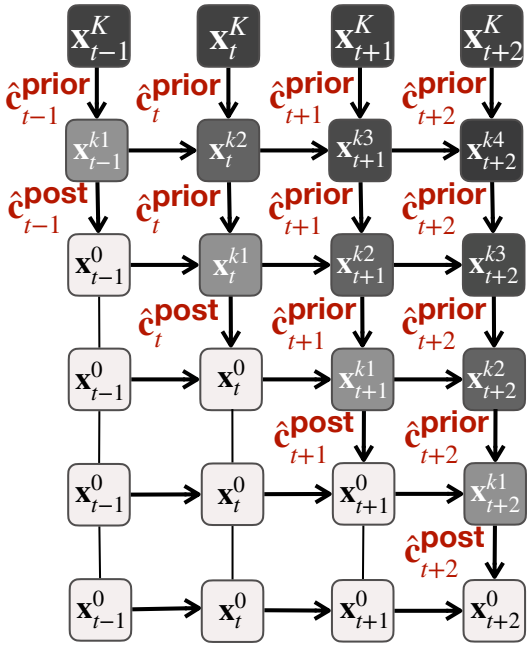

Figure A2: An illustration of the zig-zag sampling process with a block of 4 time steps. $\downarrow$ and $|$ indicate denoising and identity mapping, respectively.

D Details on `Ada-Diffuser`

D.1 Full Algorithm and Results

As illustrated in Fig. A1, our framework consists of two stages: latent factor identification and diffusion-based planning or policy learning. Below, we provide the algorithmic pseudocode for both stages. Specifically, Algorithm A1 describes Stage 1: latent factor identification, while Algorithms A2 and A3 correspond to `Ada-Diffuser-Planner` and `Ada-Diffuser-Policy`, respectively.

For clarity, we omit the detailed step-by-step procedures for denoise-and-refine and zig-zag sampling (Lines 7–8, 11, and 19–22 in Algorithm A2; Lines 6–7 and 13 in Algorithm A3), as these are fully described in Section 4.3. For `Ada-Diffuser-Policy`, we show a Diffusion Policy (DP)-based algorithm, which provides a general framework for multi-step action generation. In the IDQL-based variant, both the action execution horizon and observation horizon are set to 1, corresponding to single-step policy inference conditioned only on the current observation.

Algorithm A1: Latent Factor Identification.

1: **Input:** offline dataset $\mathcal{D}$
2: Randomly initialize decoder $p_\theta(\mathbf{s}_{t+1}, r_t \mid \mathbf{s}, \mathbf{a}, \mathbf{c})$,
encoder $q_\psi(\mathbf{c}_t \mid \mathbf{s}_{t-T_x:t+1}, \mathbf{a}_{t-T_x:t+1}, r_{t-T_x:t+1})$ and prior network $p_\phi(\mathbf{c}_t \mid \mathbf{c}_{t-1})$,
3: **while** not done **do**
4: Sample batches of trajectories from $\mathcal{D}$
5: Compute ELBO and update θ, ψ, ϕ
6: **end while**

Additionally, we provide the full results for all experiments: Table A2 reports results for the action-free setting; Tables A3 and A4 present results for environments with latent factors affecting dynamics and rewards; and Tables A5, A6, A7, and A8 summarize results for environments without explicitly modeled latent factors.

Algorithm A2: `Ada-Diffuser-Planner`.

Input: `Env`, offline dataset $\mathcal{D}$, pre-trained encoder q_ψ and prior network p_ϕ
observation horizon T_o, planning horizon T_p, action execution horizon T_a, condition $\mathbf{y}$
// Training
Initialize noise predictor ϵ_θ, inverse dynamics model f_ϕ
while not done **do**
  Sample $\mathbf{x}_{t-T_o:t+T_p}$ from $\mathcal{D}$
  Sample $\hat{\mathbf{c}}^{\text{prior}}_{t:t+T_p}$ and $\hat{\mathbf{c}}^{\text{post}}_{t:t+T_p-2}$ from p_ϕ and q_ψ
  if using inverse dynamics model **then**
    Train Causal Diffusion Model (noise predictor ϵ_θ) with $\mathbf{x}_{t-T_o:t}$, $\hat{\mathbf{c}}^{\text{prior}}_{t-T_o:t}$, and $\hat{\mathbf{c}}^{\text{post}}_{t-T_o:t}$ and other conditions $\mathbf{y}$, target outputs are $\mathbf{s}_{t+1:t+T_p}$
    Train encoder q_ψ with the contrastive improvement loss $\mathcal{L}_{\text{contrast}}$
    Train Inverse Dynamics Model f_ϕ to generate actions $\mathbf{a}_{t+1:t+T_p}$
  else
    Train Causal Diffusion Model (noise predictor ϵ_θ) with $\mathbf{x}_{t-T_o:t}$, $\hat{\mathbf{c}}^{\text{prior}}_{t-T_o:t}$, and $\hat{\mathbf{c}}^{\text{post}}_{t-T_o:t}$ and other conditions $\mathbf{y}$, target outputs are $\{\mathbf{s}_{t+1:t+T_p}, \mathbf{a}_{t+1:t+T_p}\}$
    Train encoder q_ψ with the contrastive improvement loss $\mathcal{L}_{\text{contrast}}$
  end if
end while
// Execution
Initialize environment: $s_0 \sim$ `Env.reset()`, set $t \leftarrow 0$
while not done **do**
  // Observe and infer latent factors
  Observe recent trajectory $\mathbf{x}_{t-T_o:t}$
  Sample latent variables $\hat{\mathbf{c}}^{\text{prior}}_{t:t+T_p}$ from p_ϕ
  // Generate candidate trajectory
  if using inverse dynamics model **then**
    Generate future states (zig-zag sampling) $\hat{\mathbf{s}}_{t+1:t+T_p}$ conditioned on $\mathbf{x}_{t-T_o:t}$, $\hat{\mathbf{c}}^{\text{prior}}_{t:t+T_p}$, and $\mathbf{y}$ via learned noise predictor ϵ_θ
    Generate actions $\hat{\mathbf{a}}_{t+1:t+T_p} \leftarrow f_\phi(\hat{\mathbf{s}}_{t+1:t+T_p}, \hat{\mathbf{s}}_{t:t+T_p-1})$
  else
    Generate future trajectory $\{\hat{\mathbf{s}}_{t+1:t+T_p}, \hat{\mathbf{a}}_{t+1:t+T_p}\}$ conditioned on $\mathbf{x}_{t-T_o:t}$, $\hat{\mathbf{c}}^{\text{prior}}_{t:t+T_p}$, and $\mathbf{y}$ via learned noise predictor ϵ_θ
  end if
  // Execute action(s) in environment
  for each step $i = 1$ to T_a **do**
    Execute $\hat{\mathbf{a}}_{t+i}$ in `Env`, observe s_{t+i+1}, r_{t+i}
    Append $(s_{t+i}, \hat{a}_{t+i}, r_{t+i})$ to trajectory buffer
  end for
  Update $t \leftarrow t + T_a$
end while

D.2 Architecture Choices and Hyper-parameters

We detail the architectural design choices and hyperparameter settings used for model components, loss functions, and training procedures across all `Ada-Diffuser` variants under different environments and benchmarks.

D.2.1 Latent Factor Identification

Architectures We use a variational autoencoder (VAE) [72] to optimize the evidence lower bound (ELBO). The same architectural design is used across all variants of `Ada-Diffuser` and all benchmark settings.

For the encoder, we first embed states, actions, and rewards using separate MLPs with ReLU activations. The resulting embeddings are concatenated and passed through a two-layer MLP (each

Algorithm A3: `Ada-Diffuser-Policy (DP-based)`

Input: `Env`, offline dataset $\mathcal{D}$, pre-trained encoder q_ψ and prior network p_ϕ
observation horizon T_o, action generation horizon T_p, action execution horizon T_a, condition $\mathbf{y}$
// Training
Initialize noise predictor ϵ_θ
while not done **do**
  Sample $\mathbf{x}_{t-T_o:t+T_p}$ from $\mathcal{D}$
  Sample latent variables $\hat{\mathbf{c}}^{\text{prior}}_{t:t+T_p} \sim p_\phi$, $\hat{\mathbf{c}}^{\text{post}}_{t:t+T_p-2} \sim q_\psi$
  Train causal diffusion model (noise predictor ϵ_θ) to generate actions $\mathbf{a}_{t+1:t+T_p}$, conditioned on $\mathbf{x}_{t-T_o:t}$, $\hat{\mathbf{c}}^{\text{prior}}_{t:t+T_p}$, $\hat{\mathbf{c}}^{\text{post}}_{t:t+T_p-2}$, and $\mathbf{y}$
  Train encoder q_ψ with the contrastive improvement loss $\mathcal{L}_{\text{contrast}}$
end while
// Execution
Initialize environment: $s_0 \sim$ `Env.reset()`, set $t \leftarrow 0$
while not done **do**
  // Observe and infer latent factors
  Observe recent trajectory $\mathbf{x}_{t-T_o:t}$
  Sample latent variables $\hat{\mathbf{c}}^{\text{prior}}_{t:t+T_p} \sim p_\phi$
  // Generate actions using causal diffusion model
  Generate actions (zig-zag sampling) $\hat{\mathbf{a}}_{t+1:t+T_p}$ conditioned on $\mathbf{x}_{t-T_o:t}$, $\hat{\mathbf{c}}_{t:t+T_p}$, and $\mathbf{y}$ via learned noise predictor ϵ_θ
  // Execute action(s) in environment
  for each step $i = 1$ to T_a **do**
    Execute $\hat{\mathbf{a}}_{t+i}$ in `Env`, observe s_{t+i+1}, r_{t+i}
    Append $(s_{t+i}, \hat{a}_{t+i}, r_{t+i})$ to trajectory buffer
  end for
  Update $t \leftarrow t + T_a$
end while

Environment	LDP (AF)	Ours (AF)	LDP (AF, SubOpt)	Ours (AF, SubOpt)
Lift	0.67± 0.01	**0.78**± 0.05	**1.00**± 0.00	0.98± 0.02
Can	0.78± 0.04	**0.85**± 0.07	**0.98**± 0.00	**0.98**± 0.02
Square	0.47± 0.03	**0.54**± 0.05	0.83± 0.01	**0.89**± 0.03

Table A2: **Results (success rate) on action-free demonstrations.** Here, AF and SubOpt indicate using Action-free and suboptimal demonstrations on Robomimic tasks, respectively (following the settings in LDP [77]).

layer of size 64) followed by a GRU. The GRU output is used to parameterize a Gaussian distribution from which the latent variables are sampled.

The state and reward decoders are implemented as separate MLPs, each consisting of two fully connected layers of size 64 with ReLU activations. For the prior network, we use the output of the previous step's latent distribution embedding (shared GRU) and feed it into a two-layer MLP (each layer of size 32) to predict the parameters of the prior distribution.

Loss Function At each time step t, we optimize the following losses:

$$\mathcal{L}_{\text{ELBO},t} = \underbrace{\mathbb{E}_{q_\psi(\mathbf{c}_t|\mathbf{x}_{t-T_x:t+1})}\left[-\log p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{c}_t)\right]}_{\text{Reconstruction loss}} + \underbrace{D_{\text{KL}}\left(q_\psi(\mathbf{c}_t \mid \mathbf{x}_{t-T_x:t+1}) \,\|\, p_\phi(\mathbf{c}_t \mid \mathbf{c}_{t-1})\right)}_{\text{KL regularization}}.$$

Here, $\mathbf{x}_t$ may include different components depending on the setting (e.g., $\mathbf{x}_t = \{\mathbf{s}_t, \mathbf{a}_t\}$ or $\mathbf{x}_t = \mathbf{s}_t$), and $\mathbf{c}_t$ denotes the latent context variable inferred from a temporal block of observations. The first

term encourages accurate reconstruction of the current observation $\mathbf{x}_t$ conditioned on its immediate past and the latent $\mathbf{c}_t$, while the second term regularizes the posterior to remain close to the learned prior $p_\phi(\mathbf{c}_t \mid \mathbf{c}_{t-1})$.

We implement the ELBO loss as a weighted combination of the reconstruction loss and the KL divergence:

$$\mathcal{L}_{\text{ELBO}} = \sum_{t=1}^{T-2} \left[\|\hat{\mathbf{x}}_t - \mathbf{x}_t\|_2^2 + \lambda_{\text{KL}} \cdot D_{\text{KL}} \left(q_\psi(\mathbf{c}_t \mid \mathbf{x}_{t-T_x:t+1}) \,\|\, p_\phi(\mathbf{c}_t \mid \mathbf{c}_{t-1}) \right) \right],$$

where $\hat{\mathbf{x}}_t$ is the model's reconstruction of the observation $\mathbf{x}_t$, and λ_{KL} is weighting coefficient. The reconstruction is computed using mean squared error (MSE), and the KL divergence is computed in closed form for Gaussian posteriors and priors. The hyperparameter λ_{KL} is set to be 0.1 and the learning rate is set to be $3e-4$.

D.2.2 Planner

For the planner, we consider two scenarios: (i) generating both states and actions, and (ii) generating states only. For the former, we build upon the Diffuser framework [5], which directly models full trajectories. For the latter, we adopt the Decision Diffuser (DD) framework [6], where the model generates future states and uses an inverse dynamics model to recover the corresponding actions via inverse dynamics model.

For type (i) (full state-action trajectory generation), we apply our method to the Cheetah and Ant environments. For the noise predictor, we use a 1D U-Net [80] with a kernel size of 5, channel multipliers set to (1, 2, 2, 2), and a base channel width of 32. The model is trained using the Adam optimizer [87] with a learning rate of 3×10^{-4}, a batch size of 64, and for 1 million training steps. The number of diffusion timesteps is 1000. We adopt classifier guidance (CG) [60] with gradient guidance on computed return, with a guidance scale $\omega = 1.5$. The observation horizon is set to 10 for both environments. The planning horizon T_p is set to 16 for Cheetah and 32 for Ant, with an action execution horizon of 1. These hyperparameters are kept consistent across baselines including Diffuser, DF, MetaDiffuser, and Diffuser combined with LILAC and DynaMITE for the Cheetah and Ant experiments (those in Table 1 and Appendix Table A3). For other components (e.g., VAE) in LDCQ, we employ all the hyperparameters in their original implementation [78].

For type (ii) (state-only generation with inverse dynamics), we use a Transformer-based noise predictor with a hidden dimension of 256 and a head dimension of 32. The architecture includes 2 DiT blocks for Walker, Kitchen, and Maze2D, and 8 DiT blocks for LIBERO.The model is trained using the Adam optimizer [87] with a learning rate of 3×10^{-4}, a batch size of 128, and for 1 million training steps. The number of diffusion timesteps is 500. The observation horizon is set to 4 for Kitchen, 2 for LIBERO, and 10 for the other environments. The planning horizon T_p is set to 16 for Kitchen, 10 for LIBERO, and 32 for the others. The action execution horizon is 8 for both Kitchen and LIBERO, and 10 for the remaining environments. For the inverse dynamics model, we use an MLP-based diffusion model consisting of a 3-layer MLP with 128 hidden units, preceded by a 2-layer embedding module with 64 hidden units. This model is trained for 1 million gradient steps.

For both cases, we set the coefficient of the contrastive improvement loss $\mathcal{L}_{\text{contrast}} = \max\{0, \mathcal{L}_{\text{prior}} - \mathcal{L}_{\text{post}}\}$ to 0.1. The key hyper-parameters are summarized in Table A9.

D.2.3 Policy

For the DP-based policy, we adopt the same architecture as the planner described earlier for Cheetah, Maze2D, Kitchen, Ant, and Walker. For LIBERO, we use a Transformer-based noise predictor with a decoder architecture comprising 12 layers, 12 attention heads, and a hidden embedding dimension of 768. Following DP [7], we apply dropout with a rate of 0.1 to both the input embeddings and attention weights. The number of diffusion timesteps is 500. When conditioning is used, we incorporate a Transformer encoder with 4 layers to encode the condition tokens, which include a sinusoidal timestep embedding and projected observed trajectory tokens (all mapped to the same embedding dimension). In this encoder-decoder setup, causal masking is applied to ensure autoregressive generation. In the unconditioned case, we prepend the sinusoidal timestep embedding to the input sequence and use a BERT-style encoder-only Transformer. All environments (Cheetah, Ant, Kitchen, Maze2D, Walker, and LIBERO) are trained using the AdamW optimizer with a learning rate of 10^{-4}, weight decay

10^{-3}, $\beta_1 = 0.9$, and $\beta_2 = 0.95$. Layer normalization is applied before each Transformer block for stability. The observation, planning, and action horizons follow the same settings used for the planner in each environment.

For the IDQL-based policy, we align all hyperparameters for Cheetah and Ant with the original IDQL implementation, using an observation, planning, and action horizon of 1. Hence, in IDQL-based ones, we do not consider autoregressive modeling. Similarly, for both cases, we use consider the coefficient before the contrastive improvement loss as 0.1.

D.3 Connection to Bayesian Filtering

In the absence of explicitly designed latent variables, our model can be interpreted as a form of *Bayesian filtering* [88]. Under a general formulation of the hidden Markov model (HMM) [89] with an additional latent dependency on observation ($\mathbf{c} \to \mathbf{x}$), the latent process over $\mathbf{c}$ captures the underlying stochasticity present in the demonstration data, which arises from both the environment dynamics and the behavior policy. In this view, the latent variable acts as a compact and expressive representation that summarizes the uncertainty in past observations, thereby improving the prediction of future observations. This, in turn, facilitates more robust policy learning and planning in the general settings.

E Extended Related Works

E.1 Diffusion Model-based Decision-making

Recent advances use diffusion models as the planner and policy for both reinforcement learning (RL) and imitation learning (IL). RL agent aims to learn a policy that maximizes cumulative rewards through interaction with an environment [90]. The agent observes a sequence of transitions $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$, where $\mathbf{s}_t \in \mathcal{S}$ denotes the state, $\mathbf{a}_t \in \mathcal{A}$ the action, $r_t \in \mathbb{R}$ the received reward, and $\mathbf{s}_{t+1}$ the next state. The goal is to learn a policy $\pi(\mathbf{a} \mid \mathbf{s})$ that maximizes the expected return: $\pi^* = \arg\max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where $\gamma \in [0, 1)$ is the discount factor. In contrast, IL [91] focuses on learning policies from expert demonstrations, often without access to the reward signal. A common approach is behavior cloning (BC) [92], which formulates IL as a supervised learning problem by maximizing the likelihood of expert actions given observed states, i.e., learning a policy $\pi(\mathbf{a} \mid \mathbf{s})$ that closely imitates the expert policy $\pi_e(\mathbf{a} \mid \mathbf{s})$.

Diffusion Planner Diffusion-based planning methods are commonly used to approximate the sequence of future states and actions from a given current state. By leveraging the conditional generation capabilities of diffusion models—such as guidance techniques [58, 59]—these methods can generate plans (i.e., state trajectories) that satisfy desired properties, such as maximizing expected rewards. Taking Denoising Diffusion Probabilistic Models (DDPM [60])-based approaches as an example, these methods learn a generative model over expert trajectories $\tau = \{(\mathbf{s}_0, \mathbf{a}_0), \ldots, (\mathbf{s}_T, \mathbf{a}_T)\}$ by modeling a forward-noising process: $q(\mathbf{x}^k \mid \mathbf{x}^{k-1}) = \mathcal{N}(\mathbf{x}^k; \sqrt{\alpha_k}\,\mathbf{x}^{k-1}, (1-\alpha_k)\mathbf{I})$, and a parameterized denoising model $p_\theta(\mathbf{x}^{k-1} \mid \mathbf{x}^k)$ to reverse the process. Here, k denotes the diffusion step, $\mathbf{x}^0$ is a clean sub-sequence sampled from the expert trajectory τ, and α_k controls the variance schedule at step k.

During inference, trajectories are generated by starting from Gaussian noise and iteratively denoising through the learned reverse process. This generation can be optionally conditioned on the initial state or other guidance signals $\mathbf{y}$, such as rewards, goals, or other constraints: $\hat{\tau} \sim p_\theta(\tau \mid \mathbf{s}_0, \mathbf{y})$.

These methods generally fall into two main categories: (1) learning a joint distribution over state-action trajectories, as in Diffuser [5], or (2) learning only state trajectories via diffusion and using an inverse dynamics model to recover actions, as in Decision Diffuser (DD) [6]. Beyond these, several variants extend diffusion-based planning in different directions. For example, Latent Diffuser [93] plans in a high-level latent skill space to improve generalization and LDP [77] plans with high-level latent actions directly from high-dimensional action-free demonstrations. Other approaches incorporate multi-task context to enhance adaptation and performance in unseen tasks, including MetaDiffuser [53], AdaptDiffuser [94], and MTDiff-p [95]. In addition, recent efforts have explored various extensions of diffusion planning, such as ensuring safety during generation [96], handling

multi-agent scenarios [97, 98], learning skills [99], and application in RL from human feedback (RLHF) [100].

Diffusion Policy In contrast to diffusion-based planners, Diffusion Policy methods directly parameterize the policy $\pi_\theta(\mathbf{a} \mid \mathbf{s})$ using diffusion models. For example, Diffusion Policy [7] uses a diffusion model to generate actions with expressive, multimodal distributions. DPPO [8] extends this idea by modeling a two-layer MDP structure, where the inner MDP represents the denoising process and the outer MDP corresponds to the environment. This framework enables fine-tuning of diffusion-based policies in RL settings. Another line of work integrates diffusion models with model-free methods for offline RL by using diffusion models as to model the action distributions [9–12].

Recent explorations have also aimed to unify diffusion-based planning and policy learning within a single framework. For example, the Unified Video Action model (UVA) [19] and Unified World Models (UWM) [20] leverage diffusion models to jointly model planning and action generation, demonstrating scalability on large-scale robotic tasks with pre-training. In a similar spirit, `Ada-Diffuser` provides a general framework that can be integrated into both diffusion planners and diffusion-based policies. However, `Ada-Diffuser` differs in its explicit modeling of latent factors that influence the data generation process. By incorporating latent identification directly into the diffusion process, `Ada-Diffuser` enables more structured, context-aware decision-making in partially observable and dynamically changing environments.

E.2 Latent Belief State Learning in POMDP

In partially observable Markov decision processes (POMDPs), single-step observations are typically insufficient for making optimal decisions. A common strategy to overcome this limitation involves encoding an agent's history, encoding past observations and actions into a belief state that captures a distribution over latent environmental states. Although such belief representations can, in theory, support optimal policy derivation [1, 37, 36], their exact computation depends on full knowledge of the transition and observation models. This requirement quickly becomes intractable in high-dimensional settings.

To address this, recent work has focused on learning approximate belief representations directly from data. Notable approaches include those using recurrent neural networks [38] and variational inference methods [39, 40], which enable agents to encode temporal structure and uncertainty into compact latent embeddings. These representations are then used to inform downstream policy learning, optimizing for cumulative rewards.

This direction also aligns with developments in meta-reinforcement learning and non-stationarity, where belief states or Bayesian embeddings are used to capture hidden task contexts. Agents trained across a distribution of tasks can use these latent variables to infer new environments and adapt quickly [28, 45, 43, 52, 29]. For example, MetaDiffuser [53] incorporates task context as conditioning input to diffusion-based decision models.

Our approach diverges from these by offering theoretical guarantees on the identifiability of latent factors from minimal temporal observations. Rather than depending on diverse multi-environment data, we introduce a framework that captures the full data generation process in RL using diffusion models. In contrast to MetaDiffuser, which assumes static task-level context, our model treats the latent context as a dynamic, time-evolving process that governs both environment transitions and agent behavior, capturing the underlying temporal structure of RL trajectories more faithfully.

E.3 Autoregressive Diffusion Models

To model temporal consistency and dynamics in sequential data such as videos and audios, recent work has incorporated autoregressive structures into diffusion models. These approaches differ in how they condition on prior time steps during generation and can be categorized into two main categories. **(1) Conditioning on clean (denoised) inputs** ([54–56]). At each time step t, the denoising model is conditioned on the previously denoised outputs $\{\mathbf{x}^0_{<t}\}$: $p_\theta(\mathbf{x}^{k-1}_t \mid \mathbf{x}^k_t, \mathbf{x}^0_{<t})$, where $\mathbf{x}^k_t$ is the current noisy input, and $\mathbf{x}^0_{<t}$ denotes the clean (fully denoised) observations from earlier time steps. **(2) Conditioning on noisy inputs** ([47–50]). These methods instead condition on previous time steps at their corresponding noise levels. This setting can be further divided into two cases: (a) *fully noisy conditioning* [47]: the model conditions on all prior time steps at the same noise level

k: $p_\theta(\mathbf{x}_{<t}^{k-1}, \mathbf{x}_t^{k-1} \mid \mathbf{x}_t^k, \mathbf{x}_{<t}^k,)$. (b) *partially noisy conditioning*: each previous time step $i < t$ is conditioned at its own noise level k_i, which may vary over time: $p_\theta(\mathbf{x}_0^{k_0-1}, \mathbf{x}_1^{k_1-1}, \ldots, \mathbf{x}_T^{k_T-1} \mid \mathbf{x}_0^{k_0}, \mathbf{x}_1^{k_1}, \ldots, \mathbf{x}_T^{k_T})$. Specifically, Diffusion Forcing (DF) [48] proposes a general framework in which each time step $\mathbf{x}_t$ assigns an independent noise level. In contrast, other works adopt time-dependent noise schedules that vary with the temporal index [49, 50, 57].

To model the causal generative process of RL trajectories, our approach also employs time-dependent noise scheduling to capture temporal dynamics. However, unlike prior work, we further integrate the identification of latent factors directly into the denoising process. This is achieved through a structured reinforcement step during training and a zig-zag inference procedure at test time, enabling our model to more faithfully recover the underlying causal structure in sequential decision-making.

F Benchmark Settings and Illustrations

F.1 Latent Change Factors Design

We consider the latent change factors on dynamics and rewards. We consider the Half-Cheetah and Ant environments from the OpenAI Gym suite, which are widely used MuJoCo locomotion benchmarks [101] for evaluating continuous control algorithms. In Half-Cheetah, the agent is a planar bipedal robot with a 17-dimensional state space and a 6-dimensional continuous action space, where the goal is to move forward by applying torques to six actuated joints. In Ant, a quadrupedal robot operates in a 3D space with a 111-dimensional state space and an 8-dimensional action space, requiring more complex coordination across its four legs. In both environments, the reward encourages forward velocity while penalizing excessive control inputs and, in the case of Ant, also promotes stable contact with the ground. We consider variants of the Half-Cheetah environment to study changes in dynamics, specifically **Cheetah-Wind-E** and **Cheetah-Wind-S**, which introduce external wind forces applied to the agent. In **Cheetah-Wind-E**, an opposing wind force is applied at the beginning of each episode and remains constant throughout, defined as $f_w = 10 + 5\sin(0.8i)$, where i is the episode index. For this case, since $\mathbf{c}$ change over episode, we use data from several consecutive episodes to estimate it. In **Cheetah-Wind-S**, the wind force varies at every time step according to the same formula $f_w = 5 + 5\sin(0.5t)$, with t now representing the time step in each episode. We also consider variations in the reward function. In **Cheetah-Dir-E**, the reward depends on a time-varying goal direction, requiring the agent to alternate between moving forward and backward. Specifically, the reward at episode t is defined as

$$r_t = d_t \cdot v_t - 0.1\|\mathbf{a}_t\|^2,$$

where v_t is the agent's forward velocity, $\mathbf{a}_t$ is the action vector (torques applied), and $d_t \in \{-1, +1\}$ indicates the target direction at time t. The direction signal d_t changes, giving a non-stationary reward function that challenges the policy to adapt to shifting goals. Specifically, we consider

$$d_t = \sigma(5 \cdot \sin(2\pi t/200)),$$

where $\sigma(\cdot)$ denotes the sigmoid function, α controls the sharpness of the transition, and T determines the switching period. This formulation induces a smooth periodic change in the preferred direction of movement, requiring the policy to adapt to gradually shifting objectives.

We also consider a directional reward variant for the **Ant** environment, denoted as **Ant-Dir-E**, where the agent is required to alternate its movement direction over time. The reward function at time step t is defined as

$$r_t = (2d_t - 1) \cdot v_t^x - 0.1\|\mathbf{a}_t\|^2,$$

where v_t^x is the velocity of the agent's torso along the x-axis (forward direction), $\mathbf{a}_t$ is the 8-dimensional action vector, and $d_t \in [0, 1]$ is a smooth directional signal. Similarly, we define d_t as:

$$d_t = \sigma(5 \cdot \sin(2\pi t/200)),$$

where $\sigma(\cdot)$ denotes the sigmoid function. This formulation causes the preferred movement direction to alternate approximately every 100 steps. Notably, for these settings with periodic changes (i.e., where latent factors do not evolve at every timestep), we estimate the latent variables periodically and perform refinement in the causal diffusion model only when changes are detected. This follows the same overall framework, but operates at a coarser temporal resolution aligned with the latent change frequency.

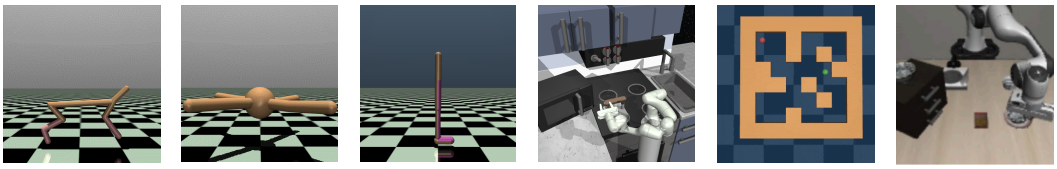

Figure A3: **Illustrations of the Benchmarks.** From left to right: Half-Cheetah, Ant, Walker, Franka-Kitchen, Maze2D, and LIBERO.

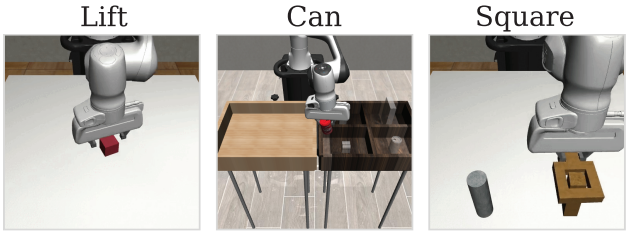

Figure A4: **Illustrations of RoboMimic Benchmark.**

F.2 Overview on Other Benchmarks

Fig. A3-A4 give the illustrations on the used benchmarks. Specifically, other than Cheetah and Ant we introduced before, for others, we consider the basic settings in offline RL. Specifically,

Maze2D. Maze2D tasks focus on goal-directed navigation in a 2D plane, where the agent must traverse a maze-like environment to reach specified targets. These settings are designed to evaluate an agent's ability to reason spatially and follow optimal trajectories based solely on positional and velocity observations.

Franka-Kitchen. The Franka-Kitchen environment [73] involves a robotic arm interacting with a series of articulated objects in a realistic kitchen setting. Tasks are composed of multiple stages, such as opening doors or toggling switches, and are intended to assess an agent's capability in handling long-horizon, multi-step manipulation.

Walker. The Walker2D environment features a two-legged robot that must learn to walk and balance using continuous torque control. The agent's objective is to maintain forward motion while remaining upright, which requires learning dynamic stability and coordination.

LIBERO [76]. The Libero benchmark offers a diverse set of continual learning tasks focused on object manipulation and generalization:

- **LIBERO-Object:** The robot is required to manipulate a variety of novel objects through pick-and-place operations. Each task introduces previously unseen objects, encouraging the agent to incrementally build knowledge about object-specific properties and behaviors.
- **LIBERO-Goal:** All tasks share a common object set and spatial layout, but vary in goal specifications. This setup tests the agent's ability to continually adapt to new task intents and motion targets without changes in the visual scene.
- **LIBERO-Spatial:** Tasks involve repositioning a bowl onto different plate locations. Although the objects remain fixed, the spatial configurations vary across tasks, requiring the robot to incrementally acquire relational spatial understanding.

RoboMimic. RoboMimic [75] provides a set of manipulation tasks based on human teleoperation demonstrations, varying in difficulty and required precision:

- **Lift:** The robot arm is tasked with lifting a small cube off the table. This task serves as a foundational manipulation scenario focused on grasping and vertical motion.

- **Can:** The robot must retrieve a cylindrical can from a cluttered bin and place it into a designated smaller container. This task introduces greater complexity due to object shape and the need for accurate placement.
- **Square:** A fine-grained insertion task where the robot picks up a square nut and places it onto a vertical rod. This is the most challenging of the three, requiring precise alignment and control for successful completion.

G Other Details on `Ada-Diffuser`

G.1 Latent Action Planner

For the latent action planner, we align our settings with those used in LDP [77], specifically focusing on learning directly from image-based demonstrations. We first use a variational autoencoder (VAE) to extract latent representations $\mathbf{z}$ from raw images via image encoders. An inverse dynamics model is then trained to recover actions $\mathbf{a}_t$ from pairs of latent states $(\mathbf{z}_t, \mathbf{z}_{t+1})$. A planner is subsequently trained to forecast future latents $\mathbf{z}$.

In our framework, we treat the latent factors $\mathbf{c}$ as high-level latent actions that influence the evolution of $\mathbf{z}$. These latent factors are jointly used with $\mathbf{z}$ to perform both inverse dynamics modeling and latent forecasting, enabling structured planning in the latent space.

We follow the experimental settings established in LDP [77]. Specifically, we use expert demonstrations alongside action-free and suboptimal demonstrations. All hyperparameters and architectural choices for the diffusion models are kept identical to those used in the original LDP implementation. We also directly utilize the pre-trained image encoder provided by LDP. The only modification in our framework is the introduction of an additional latent factor $\mathbf{c}$ trained by our latent factor identification stage, which is incorporated into the model to enhance latent action planning.

G.2 Noise Scheduling

In the autoregressive setting, we consider a monotonic increasing denoising schedule $\{k_1, \ldots, k_T\}$. In practice, we use a linear schedule where $k_i = \frac{i}{T}K$, with K denoting the maximum diffusion step used in both training and sampling. We segment the sequence into temporal blocks of length $T_x + 1$ ($T_x = T_o$ in all settings), and slide the time window forward by one step at a time. This design ensures that the denoising steps progressively increase across the block, aligning the diffusion process with the underlying temporal structure. Such a schedule encourages early steps to rely more on strong priors and later steps to refine based on more contextual information. Additionally, for better illustration, Fig. A2 provides a detailed illustration of the zig-zag sampling process within a temporal block of 4 timesteps.

H Specific Design Choices for Baselines

For all baselines, unless otherwise specified, we use the same set of diffusion parameters detailed in Appendix D.2.2–D.2.3. Below, we provide additional details on how specific methods are evaluated. While their diffusion backbones remain consistent as in Appendix D.2.2–D.2.3, these methods include custom design choices and method-specific hyperparameters that are evaluated accordingly.

H.1 Details on LILAC and DynaMITE

In these settings, we extend both LILAC and DynaMITE by incorporating a context encoder to infer latent context variables $\mathbf{c}_t$, following their respective designs. Both methods learn belief state embeddings from historical observations. For a fair comparison, we use the same latent identification network architecture as in our framework, but modify the inputs according to each method's assumptions.

Specifically, LILAC and DynaMITE condition their inference networks solely on the historical trajectory $\mathbf{x}_{1:t}$, without access to current and future information. Additionally, consistent with their original implementations, we do not include a separate prior head on top of the GRU; both methods share the encoder for posterior inference and prior prediction. And the primary difference

(in implementation) between these two methods lies in the temporal context used: LILAC maintains the full belief over the entire history, i.e., it conditions on $\mathbf{x}_{1:t}$ to infer $\mathbf{c}_{t+1}$, while DynaMITE uses only the most recent context, i.e., it infers $\mathbf{c}_{t+1}$ based solely on $\mathbf{x}_t$.

All other hyperparameters are aligned with those used in our Stage 1 training. The estimated context variables are then provided as additional conditioning inputs to the diffusion-based models.

H.2 Details on Diffusion Forcing

For Diffusion Forcing, we adopt the same autoregressive noise schedule as in our method, which accounts for causal uncertainty, similarly to the formulation in Eq. D.1 of [48], to ensure a fair comparison. Additionally, we use the Monte Carlo Guidance (MCG) mechanism introduced in [48] for Maze2D, following the original setup. For all other environments, we use the same classifier guidance scheme as the other baselines to maintain consistency in evaluation.

I Ablation Analysis

I.1 Ablation Results

I.1.1 Full Results Supplement to Table 3

Table A10 presents the full ablation results across all environments, as a supplement to Table 3. Overall, the results highlight the importance of the two key components in causal diffusion modeling: latent identification and autoregressive diffusion, both of which are critical for performance.

I.1.2 Effect of Temporal Block Length on Latent Identification

We further analyze the impact of temporal block length on latent identification. As shown in Fig. A5, the results are consistent with findings reported in the main paper. When the number of observations is insufficient (e.g., ≤ 4), identification performance degrades. Performance improves when the block length is in a moderate range (5–20), indicating that sufficient temporal context is beneficial. However, using overly long blocks (> 20) introduces redundancy and increases optimization difficulty, which in turn harms performance.

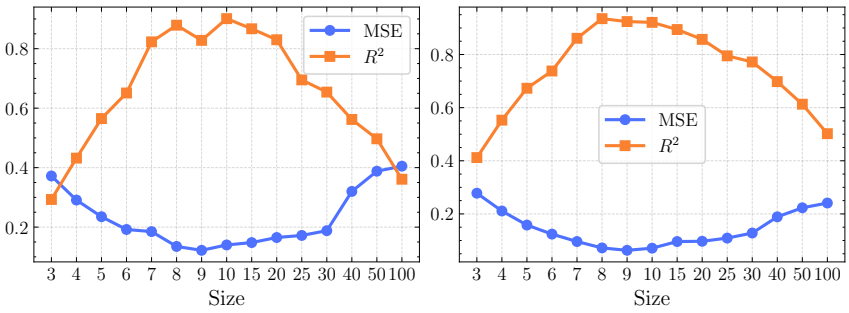

Figure A5: **Identification results (MSE of linear probing and R^2) versus the length of temporal blocks.** Left: Cheetah with time-varying wind; Right: Cheetah with time-varying rewards.

I.1.3 On the Effect of Planning and Execution Horizons: Long-horizon Planning

We study the robustness of our approach under increased planning and execution horizons (T_p and T_a). Specifically, we evaluate on two challenging tasks—Franka-Kitchen-Partial and Libero-Long, where the original settings are Kitchen ($T_p = 16$, $T_a = 8$) and Libero ($T_p = 10$, $T_a = 8$). Results are in Fig. A6. When we increase these horizons, we observe that the baselines, DP and DF, suffer significant performance drops. In contrast, `Ada-Diffuser`maintains relatively high performance. This demonstrates that modeling the underlying causal generative process, through autoregressive structure and latent representations, enables better **long-horizon planning**. Although we do not explicitly impose latent variables, our model implicitly learns representations that can track stochasticity and support smooth control.

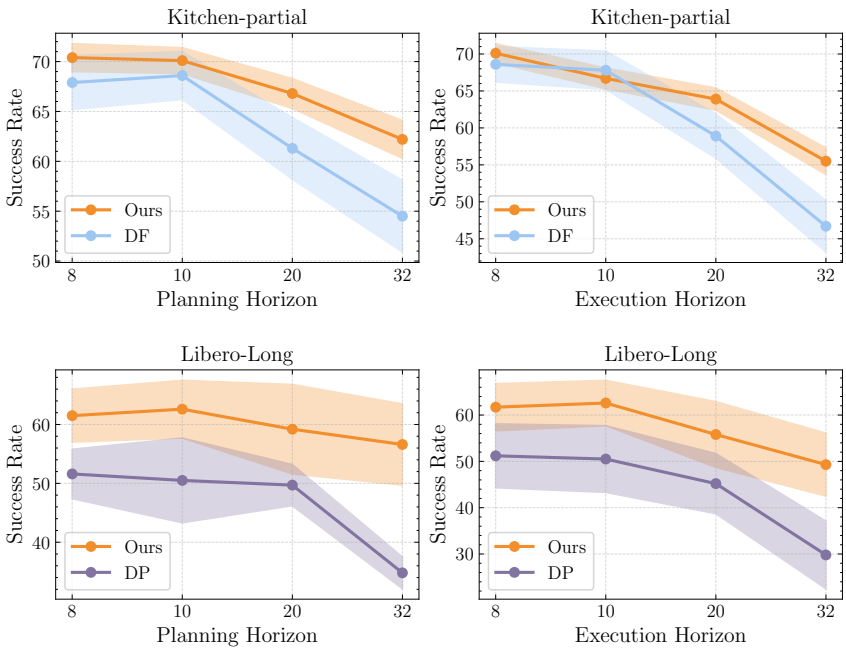

Figure A6: **Results with different planning and execution horizons.** We evaluate on Kitchen-partial and Libero-Long experiments.

I.2 Training/Inference Time Analysis

We conduct all experiments on 4x NVIDIA A100 or 8× RTX 4090 GPUs, depending on the model scale and environment requirements. The main sources of computational overhead in our framework arise from two components: (i) the latent factor identification network, and (ii) the denoise-and-refine steps in the diffusion model. During sampling, the overhead comes from both zig-zag sampling and latent variable sampling. However, these additions do not introduce significant overhead in either training or inference time. To quantify this, we report the training and testing speed of our method compared to the base models DD and DP, across all environments. As shown in Table A11, our method introduces only a moderate computational overhead, typically around 20–30% more training and inference time compared to baseline methods. We believe this overhead can be further optimized through techniques such as parallel latent sampling, more lightweight context encoders, or only inference-time refinement. Moreover, as demonstrated in prior sections, `Ada-Diffuser` is capable of handling long-horizon tasks, offering a favorable trade-off between long-horizon planning and inference efficiency.

Environment	Diffuser	DF	MetaDiffuser	Diffuser + DynaMITE	Diffuser + LILAC	Ours	Ours + Meta	Oracle
Cheetah-Wind-E ($\mathbf{c}^s$)	-120.4 $\pm$ 12.7	-105.8 $\pm$ 9.6	-89.7 $\pm$ 6.5	-79.2 $\pm$ 11.0	-95.3 $\pm$ 7.4	**-68.9** $\pm$ 7.6	-62.4 ± 3.9	-57.8 ± 6.7
Cheetah-Wind-S ($\mathbf{c}^s$)	-148.5 $\pm$ 9.8	-102.0 $\pm$ 10.2	-106.8 $\pm$ 11.4	-94.3 $\pm$ 9.6	-105.6 $\pm$ 14.5	**-73.5** $\pm$ 8.7	-65.3 $\pm$ 11.2	-58.1 $\pm$ 9.0
Cheetah-Dir-E ($\mathbf{c}^r$)	850.8 $\pm$ 54.2	902.1 $\pm$ 45.8	912.5 $\pm$ 37.9	930.4 $\pm$ 29.5	908.5 $\pm$ 37.6	**943.3** $\pm$ 25.6	949.8 $\pm$ 24.1	962.1 $\pm$ 21.9
Cheetah-Vel-E ($\mathbf{c}^r$)	-102.4 $\pm$ 18.2	-85.6 $\pm$ 18.3	-69.2 $\pm$ 7.5	-76.3 $\pm$ 11.7	-62.6 $\pm$ 11.1	**-45.8** $\pm$ 9.5	-39.2 $\pm$ 7.6	-38.3 $\pm$ 8.9
Ant-Dir-E ($\mathbf{c}^r$)	188.6 $\pm$ 39.2	195.4 $\pm$ 47.0	245.9 $\pm$ 41.0	262.8 $\pm$ 27.5	229.4 $\pm$ 32.6	**285.3** $\pm$ 24.5	296.4 $\pm$ 22.2	300.7 $\pm$ 23.6

Table A3: **Results (average returns) on `Ada-Diffuser`-Planner with latent factors that affects dynamics and rewards.** $\mathbf{c}^s$ and $\mathbf{c}^r$ indicate the changes on dynamics and reward, E and S represent the episodic and time-step changes. The results are with 5 random seeds. The bold ones are the best-performing ones, excluding meta-learning and oracle ones.

Environment	DP	DP + DynaMITE	Ours + DP	Ours + DP (Oracle)	IDQL	IDQL + DynaMITE	Ours + IDQL	Ours + IDQL (Oracle)
Cheetah-Wind-E ($\mathbf{c}^s$)	-104.8 $\pm$ 10.9	-72.2 $\pm$ 5.9	**-58.5** $\pm$ 4.6	-52.0 $\pm$ 3.5	-97.5 $\pm$ 9.4	-59.0 $\pm$ 11.2	**-48.5** $\pm$ 7.9	-41.6 $\pm$ 6.2
Cheetah-Wind-S ($\mathbf{c}^s$)	-120.6 $\pm$ 11.5	-76.5 $\pm$ 15.6	**-52.9** $\pm$ 9.8	-42.3 $\pm$ 6.7	-87.8 $\pm$ 12.2	-63.4 $\pm$ 6.7	**-48.0** $\pm$ 7.2	-44.7 $\pm$ 6.1
Cheetah-Dir-E ($\mathbf{c}^r$)	892.5 $\pm$ 60.8	949.6 $\pm$ 36.1	**960.7** $\pm$ 40.2	972.4 $\pm$ 37.5	902.4 $\pm$ 45.2	938.6 $\pm$ 49.4	**965.0** $\pm$ 37.5	969.8 $\pm$ 39.2
Cheetah-Vel-E ($\mathbf{c}^r$)	-87.9 $\pm$ 6.5	-72.7 $\pm$ 5.8	**-41.0** $\pm$ 7.2	-39.8 $\pm$ 6.7	-80.2 $\pm$ 11.4	-59.4 $\pm$ 6.5	**-38.6** $\pm$ 7.7	-33.8 $\pm$ 6.5
Ant-Dir-E ($\mathbf{c}^r$)	182.5 $\pm$ 41.2	275.2 $\pm$ 27.0	**290.4** $\pm$ 49.4	312.5 $\pm$ 37.2	204.6 $\pm$ 25.6	269.3 $\pm$ 29.5	**295.8** $\pm$ 32.7	309.6 $\pm$ 25.4

Table A4: **Results (average returns) on `Ada-Diffuser`-Policy with latent factors.** $\mathbf{c}^s$ and $\mathbf{c}^r$ indicate the changes on dynamics and reward, E and S represent the episodic and time-step changes. The results are with 5 random seeds. The bold ones are the best-performing ones, excluding meta-learning and oracle ones.

Environment	Diffuser	DD	DF	LDCQ	Ours (DD)
Mixed	52.6 ± 2.3	**75.2** ± 1.4	73.7 ± 1.9	73.3 ± 0.5	74.6± 1.6
Partial	55.8± 1.9	57.3± 1.2	68.6± 2.4	67.8± 0.8	**70.1**± 1.3

Table A5: **Results (success rate (%)) on `Ada-Diffuser`-Planner without explicit latent factors on Franka-kitchen environment.** The results are with 5 random seeds.

Environment	Diffuser	DD	DF	LDCQ	Ours (DD)
umaze	113.5 ± 2.8	114.8 ± 3.2	116.7 ± 2.0	134.2 ± 4.1	**148.6**± 3.7
medium	121.5 ± 5.6	129.6 ± 2.9	**149.4**± 7.5	125.3± 2.5	148.6± 3.1
large	123.0± 4.8	131.5± 4.2	159.0± 2.7	150.1± 2.9	**161.4**± 3.2

Table A6: **Results on `Ada-Diffuser`-Planner without explicit latent factors on Maze-2D environment**. The results are averaged across 5 random seeds.

Environment	Diffuser	DD	DF	LDCQ	Ours (DD)
medium-expert	106.2 ± 0.7	108.8 ± 2.0	105.4 ± 3.2	109.3 ± 0.4	**115.7** ± 2.1
medium	79.6± 9.8	82.5± 1.6	66.2± 1.9	69.4± 2.4	**83.6**± 3.5
medium-replay	70.6± 0.6	75.0 ± 3.2	72.2± 2.6	68.5± 4.3	74.3± 2.8

Table A7: **Results on `Ada-Diffuser`-Planner without explicit latent factors on Walker environment**. The results are averaged across 5 random seeds.

Environment	DP	Ours (DP)
Spatial	78.3 ± 3.9	**79.2** ± 4.2
Object	92.5± 2.6	**93.4**± 2.8
Long	50.5 ± 7.2	**62.6** ± 4.9

Table A8: **Results on `Ada-Diffuser`-Policy without explicit latent factors on Libero environment**. The results are averaged across 5 random seeds.

Component	Type (i): Full Trajectory	Type (ii): State-Only
Model Backbone	1D U-Net [80]	Transformer (DiT)
Architecture	Kernel size: 5; channels: (1,2,2,2); base: 32	Hidden dim: 256; head dim: 32
# DiT Blocks	–	2 (Walker, Kitchen, Maze2D), 8 (LIBERO)
Optimizer	Adam, lr = 3×10^{-4}	Adam, lr = 3×10^{-4}
Batch Size	64	128
Training Steps	1M	1M
Diffusion Timesteps	1000	500
Observation Horizon T_o	10	4 (Kitchen), 2 (LIBERO), 10 (others)
Planning Horizon T_p	16 (Cheetah), 32 (Ant)	16 (Kitchen), 10 (LIBERO), 32 (others)
Execution Horizon T_o	1	8 (Kitchen, LIBERO), 10 (others)
Guidance	CG, $\omega = 1.5$	CFG
Inverse Dynamics Model	–	2-layer embed (64), 3-layer MLP (128), 1M steps
Refinement Loss Cofficient	0.1	0.1

Table A9: Planner configurations for type (i): full trajectory generation and type (ii): state-only generation with inverse dynamics.

Cases	Cheetah-1	Cheetah-2	Ant	Maze2D	Walker	Kitchen	RoboMimic	LIBERO
Original	**-73.5**	**-52.9**	**295.8**	**161.4**	**115.7**	**0.70**	**0.85**	**93.4**
w/o refine	-82.0	-60.7	261.2	156.5	107.4	0.63	0.78	90.2
w/o zig-zag	-91.6	-56.1	258.3	147.6	107.9	0.59	0.75	91.6
same NS	-89.7	-62.4	259.7	140.1	105.8	0.56	0.72	85.2
random NS	-84.6	-62.9	266.4	146.3	109.1	0.61	0.76	88.5

Table A10: **Ablation on Design Choices.** We conduct ablation studies across a diverse set of tasks, including: Cheetah-Wind-S with a planner-based approach (denoted as Cheetah-1 in the table), Cheetah-Wind-S with a diffusion policy (Cheetah-2), Ant-Dir-E (policy, IDQL-based), Maze2D-Large (planner), Walker2D-Medium-Expert (planner), Kitchen-Partial (planner), LIBERO-Object (diffusion policy), and RoboMimic-Can.

Environment	**Training Time (sec/epoch)**		**Inference Time (sec/rollout)**	
	Ours vs DD	Ours vs DP	Ours vs DD	Ours vs DP
Cheetah	72.1 / 60.1 (1.20)	69.8 / 58.4 (1.20)	1.51 / 1.26 (1.20)	1.57 / 1.22 (1.29)
Ant	79.5 / 64.3 (1.24)	76.0 / 62.0 (1.23)	1.67 / 1.36 (1.23)	1.72 / 1.32 (1.30)
Walker	85.3 / 67.1 (1.27)	81.5 / 64.2 (1.27)	1.83 / 1.45 (1.26)	1.88 / 1.38 (1.36)
Maze2D	90.2 / 72.0 (1.25)	88.3 / 69.2 (1.28)	1.94 / 1.54 (1.26)	2.01 / 1.46 (1.38)
Libero	104.0 / 81.0 (1.28)	102.1 / 78.0 (1.31)	2.18 / 1.72 (1.27)	2.31 / 1.64 (1.41)
Kitchen	117.8 / 88.1 (1.34)	115.3 / 85.0 (1.36)	2.45 / 1.83 (1.34)	2.55 / 1.73 (1.47)

Table A11: Training and inference time comparison for `Ada-Diffuser-planning` and `Ada-Diffuser-policy` variants. We report absolute times (in seconds per epoch/rollout) and relative overheads.